

UNIVERSIDAD DE VALLADOLID



E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE  
TELECOMUNICACIÓN

**Configuración de la red residencial del usuario  
para la integración de SDN en redes de acceso  
PON**

Autor:

**D. Alfredo González Fresnillo**

Tutor:

**Dña. Noemí Merayo Álvarez**



---

**TÍTULO: Configuración de la red residencial del usuario para la integración de SDN en redes de acceso PON**  
**AUTOR: D. Alfredo González Fresnillo**  
**TUTOR: Dña. Noemí Merayo Álvarez**  
**DEPARTAMENTO: Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

---

---

**TRIBUNAL**

**PRESIDENTE: Ignacio de Miguel Jiménez**  
**SECRETARIO: Noemí Merayo Álvarez**  
**VOCAL: Ramón J. Durán Barroso**  
**SUPLENTE: Patricia Fernández del Reguero**  
**SUPLENTE: J. Carlos Aguado Manzano**

---

---

**FECHA:**  
**CALIFICACIÓN:**

---

## **Resumen de TFG**

Este Trabajo de Fin de Grado se ha desarrollado en el laboratorio de Comunicaciones Ópticas (2L007) de la Escuela Técnica Superior de Ingenieros de Telecomunicación sobre la maqueta de red GPON que allí se encuentra. Sobre esta red está implementada una solución SDN a través del protocolo OpenFlow.

En primer lugar, se llevará a cabo un proceso de análisis del rendimiento de los dispositivos y ordenadores conectados a la red y tras analizar los resultados obtenidos se realizarán los cambios oportunos que optimicen el funcionamiento de la red, actualizando de este modo su topología. Posteriormente, se instalarán y probarán nuevos dispositivos en la red GPON y se establecerán las reglas para ampliaciones futuras.

Tras esta actualización, se analizará el flujo del programa que gestiona el entorno SDN-GPON desplegado en busca de aquellos puntos en los que el programa necesita ser automatizado de modo que se pueda aumentar la escalabilidad de la red, para posteriormente desarrollar las funciones en Python necesarias que consigan dicho objetivo.

Por último, tras comprobar que las funciones desarrolladas cumplen su función, se hará un análisis de una de las principales funcionalidades de las que cuenta el programa de gestión del entorno SDN-GPON, en concreto un algoritmo de configuración dinámica de servicios para ofrecer nuevos modelos de negocio para operadores de red y proveedores de servicio.

## **Palabras clave**

SDN (*Redes Diseñadas por Software*), GPON (*Red Óptica Pasiva con Capacidad de Gigabit*), OLT (*Terminación Óptica de Línea*), ONU/ONT (*Unidad de Red Óptica/Terminal de Red Óptica*), OVS (*Open Virtual Switch*), OpenFlow, OpenDayLight, Python, Wireshark, iperf.

## **Abstract**

This Final Degree Project has been carried out in the Optical Communications laboratory (2L007) of the ETSIT on the GPON network model that is located there. An SDN solution is implemented on this network through the OpenFlow protocol.

First, a performance analysis process of the devices and computers connected to the network will be carried out and after analyzing the results obtained, appropriate changes will be made to optimize the operation of the network, thus updating its topology. Later, new devices will be installed and tested and the rules for future extensions will be established.

After this update, the flow of the program that manages the SDN-GPON network will be analyzed in search of those points where the program needs to be automated so that the scalability of the network can be increased, to later develop the necessary Python functions that they achieve this objective.

Finally, after verifying that the developed functions fulfill their function, an analysis will be made of one of the main functions of the program, the algorithm of dynamic configuration of services, so that an explanation is given to those points in the that their behavior is not expected.

## **Keywords**

SDN (*Software Defined Networks*), GPON (*Gigabit-capable Passive Optical Networks*), OLT (*Optical Line Termination*), ONU/ONT (*Optical Network Unit/Optical Network Terminal*), OVS (*Open Virtual Switch*), OpenFlow, OpenDayLight, Python, Wireshark, iperf.



# Agradecimientos

*A mis padres y mi tío, porque gracias a ellos soy lo que soy hoy en día.*

*A Noemí, por su ayuda, compromiso y comprensión en todo momento.*

*A David, por estar siempre dispuesto a ayudar.*

*La investigación desarrollada en este Trabajo Fin de Grado ha sido financiada por el Ministerio de Ciencia, Innovación y Universidades en el marco del proyecto ONOFRE-2 (TEC2017-84423-C3-1-P) y la red de investigación Go2Edge (RED2018-102585-T), por la Consejería de Educación de la Junta de Castilla y León en el marco del proyecto ROBIN (VA085G19), y por el Fondo Europeo de Desarrollo Regional FEDER a través del proyecto DISRUPTIVE del Programa Interreg V-A España-Portugal (POCTEP) 2014-2020. Las opiniones son de exclusiva responsabilidad del autor que las emite.*

# Índice

<b>Agradecimientos</b> .....	<b>vii</b>
<b>Índice</b> .....	<b>ix</b>
<b>Índice de figuras</b> .....	<b>xi</b>
<b>Índice de tablas</b> .....	<b>xv</b>
<b>1 Introducción</b> .....	<b>1</b>
1.1 Motivación.....	1
1.2 Objetivos.....	2
1.2.1 Objetivo General.....	2
1.2.2 Objetivos Específicos .....	2
1.3 Fases y Métodos .....	3
1.3.1 Fase de Análisis .....	3
1.3.2 Fase de Implementación .....	3
1.3.3 Fase de Pruebas.....	4
1.3.4 Fase de Realización de los Informes .....	4
1.4 Estructura de la Memoria del TFG .....	4
<b>2 Metodología y herramientas de trabajo</b> .....	<b>5</b>
2.1 Introducción.....	5
2.2 Redes de Acceso Ópticas Pasivas (PON, <i>Passive Optical Networks</i> ).....	5
2.3 Introducción a SDN (Software Defined Networking) .....	7
2.4 Herramientas y software SDN utilizados .....	8
2.4.1 Protocolo OpenFlow 1.3.....	8
2.4.2 Integración de switches virtuales OVS (Open Virtual Switch).....	8
2.4.3 Controlador SDN OpenDayLight (ODL) .....	9
2.5 Lenguaje de programación Python .....	10



---

2.6	Metodología de trabajo .....	10
2.6.1	Integración y actualización de nuevos dispositivos PON.....	11
2.6.2	Integración y mejora de políticas SDN y automatización de la red .....	11
2.7	Conclusiones.....	12
<b>3</b>	<b>Actualización del entorno SDN-GPON .....</b>	<b>13</b>
3.1	Introducción.....	13
3.2	Descripción de la solución SDN-GPON inicial.....	13
3.3	Análisis del rendimiento de las tarjetas de red de los ordenadores que componen el escenario SDN-GPON .....	18
3.3.1	Diseño del escenario de red para analizar el rendimiento .....	18
3.3.2	Análisis del rendimiento de las tarjetas en el ordenador central del escenario SDN-GPON .....	23
3.3.3	Análisis del rendimiento de las tarjetas en los ordenadores nuevos .....	26
3.3.4	Configuración del nuevo escenario de red SDN-GPON .....	27
3.4	Activación de nuevos puertos PON en el OLT .....	31
3.4.1	Registro de las ONTs en puertos PON mediante TGMS .....	31
3.4.2	Prueba de funcionamiento y análisis de resultados .....	35
3.5	Registro y activación de ONTs de nivel 2 .....	38
3.5.1	Proceso de activación y análisis de rendimiento de ONTs de nivel 2 .....	39
3.5.2	Proceso de despliegue de OVS's en máquinas virtuales .....	42
3.6	Conclusiones.....	47
<b>4</b>	<b>Automatización de la red SDN-GPON .....</b>	<b>48</b>
4.1	Introducción.....	48
4.2	Descripción del flujo del programa que controla la red SDN-GPON .....	48
4.2.1	Descripción y diagrama de flujo del algoritmo de configuración dinámica de servicios en la red SDN-GPON .....	50
4.2.2	Descripción y diagrama de flujo de la aplicación para la gestión de servicios temporales.....	54

---

4.2.3	Descripción y diagrama de flujo de la aplicación para la gestión de la red residencial .....	55
4.3	Descripción de las funciones para la automatización de la red SDN-GPON ...	58
4.3.1	Automatización del puerto conectado del OLT .....	58
4.3.2	Automatización de la obtención de listas con los identificadores de los OVSs, sus interfaces y los puertos que utilizan .....	59
4.3.3	Automatización de la asociación de la dirección MAC de la interfaz del OVS con el identificador de la ONT .....	60
4.3.4	Automatización en la creación de subredes asociadas a los servidores DHCP de las ONTs.....	61
4.3.5	Creación de los flows por defecto en el controlador .....	62
4.3.6	Funcionamiento de la red residencial utilizando varias ONTs simultáneamente .....	63
4.4	Análisis del comportamiento de políticas de configuración de servicios en tiempo real .....	67
4.5	Conclusiones.....	82
<b>5</b>	<b>Conclusiones y líneas futuras.....</b>	<b>84</b>
5.1	Conclusiones.....	84
5.2	Líneas futuras.....	85
<b>6</b>	<b>Bibliografía .....</b>	<b>87</b>
<b>Anexo I</b>	<b>.....</b>	<b>90</b>
	Funciones para la automatización del código .....	90

# Índice de figuras

Figura 1: Ejemplo de topología de redes SDN .....	7
Figura 2: Interfaz web del controlador ODL .....	17
Figura 3: Aspecto real de la red de acceso GPON desplegada en el laboratorio.....	15
Figura 4: Esquema del escenario SDN-GPON desplegado sobre la red GPON tradicional .....	17
Figura 5: Arquitectura para comprobar el rendimiento de las tarjetas de red de un ordenador .....	19
Figura 6: Asignación estática de direcciones IP en los ordenadores.....	20
Figura 7: Ejemplo de salida del comando route -n .....	21
Figura 8: Gráfica ejemplo con Wireshark .....	23
Figura 9: Topología de red del ordenador central .....	23
Figura 10: Código del archivo activarServidor.sh .....	24
Figura 11: Arquitectura de la red para el escenario 1 .....	24
Figura 12: Arquitectura de la red para el escenario 2.....	24
Figura 13: Arquitectura de la red para el escenario 3 .....	25
Figura 14: Arquitectura de red para analizar el rendimiento de las tarjetas del ordenador nuevo.....	26
Figura 15: Topología de red final en el ordenador central .....	28
Figura 16: Configuración final del archivo donde se definen las interfaces y las VLANs .....	29
Figura 17: Configuración final del archivo activar-enrutamiento .....	29
Figura 18: Configuración final de red de la máquina virtual del controlador .....	30
Figura 19: Configuración final de red de la máquina virtual de TGMS.....	30
Figura 20: Imagen frontal real del OLT y los puertos PON asociados .....	31
Figura 21: Acceso en TGMS a los detalles del puerto 1 del OLT.....	32

Figura 22: Pestaña con las ONUs que están conectadas en la red PON (Disabled ONUs) pero no asociadas al puerto Port 1 .....	32
Figura 23: Ventana para configurar la ONU “ONU Configuration” .....	33
Figura 24: Estado inicial en el que aparecen las ONUs conectadas, estado Pending.....	34
Figura 25: OLT pendiente de sincronización debido al registro de nuevas ONUs .....	34
Figura 26: Captura de pantalla donde se observan todas las ONUs que están Online .....	35
Figura 27: Topología prueba del Puerto 1 del OLT .....	36
Figura 28: Captura de pantalla del interfaz de configuración del router de la ONT (WAN Info) .....	37
Figura 29: Código del archivo configuracionVLAN.sh .....	39
Figura 30: Código del archivo script.sh.....	40
Figura 31: Código del archivo script2.sh.....	40
Figura 32. Topología para la prueba de las ONTs de nivel 2. ....	41
Figura 33: Esquema de la máquina virtual del ordenador de la Torre.....	42
Figura 34: Proceso para exportar un servicio virtualizado de una máquina virtual .....	43
Figura 35: Archivo configuracionVLAN.sh para el ordenador nuevo .....	44
Figura 36: Adaptador de red 1 para la máquina virtual del nuevo OVS .....	44
Figura 37: Adaptador de red 2 para la máquina virtual del nuevo OVS .....	44
Figura 38: Archivo interfaces de la máquina virtual del nuevo OVS.....	45
Figura 39: Servidor DHCP del nuevo OVS.....	46
Figura 40: Archivo script.sh del nuevo OVS.....	46
Figura 41: Menú inicial con las funcionalidades implementadas.....	49
Figura 42: Menú Openflow Configuration .....	50
Figura 43: Diagrama de flujo del algoritmo de configuración dinámica de servicios.....	53
Figura 44: Diagrama de flujo de la aplicación para la gestión de servicios temporales...	55
Figura 45: Diagrama de flujo de la aplicación para la gestión de la red residencial .....	58
Figura 46: Código de la función getONU().....	61
Figura 47: Menú Openflow modificado .....	64

Figura 48: Contratación de un servicio temporal .....	64
Figura 49: Captura de Wireshark del servicio temporal contratado .....	65
Figura 50: Dispositivos registrados en la aplicación web .....	65
Figura 51: Restricción de un ancho de banda para un dispositivo .....	66
Figura 52: Captura de Wireshark de la restricción de ancho de banda a un dispositivo de la red residencial .....	66
Figura 53: Comportamiento del algoritmo cuando el tamaño de ventana es igual a 20 muestras .....	68
Figura 54: Comportamiento del algoritmo cuando el tamaño de ventana es igual a 15 muestras .....	68
Figura 55: Comportamiento del algoritmo cuando el tamaño de ventana es igual a 10 muestras .....	68
Figura 56: Escalones con un comportamiento inadecuado del algoritmo .....	70
Figura 57: Comportamiento de la red SDN-GPON ante un cambio en la tasa de transmisión.....	71
Figura 58: Servicio temporal a una tasa de transmisión de 180 Mbps. ....	72
Figura 59: Servicio temporal a una tasa de transmisión de 150 Mbps .....	72
Figura 60: Servicio temporal a una tasa de transmisión de 100 Mbps .....	72
Figura 61: Tarifa básica de 150 Mbps con un ancho de banda flexible asociado a un 30% .....	74
Figura 62: Tarifa básica de 100 Mbps con un ancho de banda flexible asociado del 20% .....	75
Figura 63: Tarifa de 100 Mbps con una flexibilidad del 20% y saltos de 10 Mbps.....	75
Figura 64: Ancho de banda demandado para la tarifa de 100 Mbps con un ancho de banda extra del 50%.....	76
Figura 65: Tarifa de 100 Mbps con un ancho de banda extra del 50% .....	77
Figura 66: Ancho de banda demandado para la tarifa flexible con un ancho de banda básico de 50 Mbps con un extra del 100%. ....	77
Figura 67: Tarifa flexible con un ancho de banda básico de 50 Mbps con un extra del 100% .....	78
Figura 68: Escenario real para tarifas públicas.....	79
Figura 69: Ancho de banda demandado para la tarifa pública flexible de 50 Mbps .....	79

---

Figura 70: Comportamiento del algoritmo para la tarifa pública flexible de 50 Mbps ....	80
Figura 71: Evolución de la media de la ventana para la tarifa pública flexible de 50 Mbps .....	80
Figura 72: Ancho de banda demandado para la tarifa pública flexible de 100 Mbps .....	81
Figura 73: Comportamiento del algoritmo para la tarifa pública flexible de 100 Mbps ..	82
Figura 74: Evolución de la media de la ventana para la tarifa pública flexible de 100 Mbps .....	82

## Índice de tablas

Tabla 1. Resultados de las pruebas realizadas del rendimiento de las tarjetas del ordenador central. ....	25
Tabla 2. Resultados de las pruebas realizadas del rendimiento de las tarjetas del ordenador nuevo. ....	27
Tabla 3. Resultados del ancho de banda ofrecido a la misma ONT cuando se conecta a ambos puertos para el mismo servicio y perfil de abonado asociado. ....	38
Tabla 4. Resultados de rendimiento de las ONTs de nivel 2 conectadas a la red SDN-GPON. ....	41
Tabla 5. Tipos de tarifas flexibles definidas .....	73

# 1

## Introducción

### 1.1 Motivación

Este trabajo de fin de grado, desarrollado sobre la red de acceso GPON (*Gigabit-capable Passive Optical Networks*) que se encuentra en el laboratorio 2L007 de la escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad de Valladolid se divide en dos partes bien diferenciadas.

La primera parte, trata sobre la actualización del escenario SDN (*Software Defined Networking*) desplegado en trabajos previos sobre un testbed de una red GPON, analizando aquellos puntos susceptibles de mejora, para posteriormente realizar los cambios que se consideren oportunos configurando un nuevo escenario de red que permita un funcionamiento más óptimo de todos los servicios configurados.

En la segunda parte se abordará la automatización de la aplicación desarrollada en Python en trabajos anteriores, que controla la gestión de la solución SDN-GPON implementada. Para ello, en primer lugar, se realizará un análisis del funcionamiento de las principales funcionalidades del programa de gestión en busca de aquellos puntos que puedan ser automatizados y una vez encontrados dichos puntos buscar soluciones de modo que finalmente la aplicación quede automatizada lo máximo posible.

Después de realizar esta automatización la red contará con un mayor grado de escalabilidad y flexibilidad, pudiendo instalar nuevos dispositivos al entorno SDN-GPON sin necesidad de grandes cambios en el código Python.

Finalmente, hecho todo esto, se procederá a hacer un análisis en profundidad del algoritmo de configuración dinámica de servicios también implementado en trabajos previos, buscando mejorar su funcionamiento y sus funcionalidades.



## **1.2 Objetivos**

### **1.2.1 Objetivo General**

Los objetivos principales de este Trabajo de Fin de Grado, son dos, en primer lugar, la actualización del escenario SDN-GPON, buscando optimizar lo máximo posible su funcionamiento, integrando nuevos dispositivos a la red GPON, pero intentando que mejoren las prestaciones de la red.

Por otro lado, se pretende en este TFG dotar a la red de un mayor grado de escalabilidad, mediante la automatización de la aplicación desarrollada en Python y las funcionalidades implementadas que gestionan el escenario SDN-GPON desplegado y estableciendo las reglas de configuración de nuevos dispositivos y del software que los acompaña.

Para la consecución de los objetivos que se acaban de describir, previamente es necesario analizar la configuración de la red de acceso y entender el funcionamiento del programa que gestiona el escenario de red SDN-GPON desplegado.

Para alcanzar estos objetivos generales, es necesario establecer otros objetivos más específicos, que se van a describir a continuación.

### **1.2.2 Objetivos Específicos**

Durante este Trabajo de Fin de Grado se han desarrollado los objetivos específicos que se enumeran a continuación:

1. Analizar la topología y configuración del escenario de red SDN-GPON y las funcionalidades existentes al inicio de este trabajo.
2. Estudiar las prestaciones de los ordenadores disponibles en el laboratorio en busca de que ofrezcan un comportamiento más óptimo para la gestión SDN del testbed GPON.
3. Instalar y probar nuevos dispositivos PON disponibles en el laboratorio, esto es, dos nuevas ONTs (*Optical Network Terminals*) de nivel 2 y otro puerto PON del OLT (*Optical Line Terminal*).

4. Establecer las reglas para la instalación de nuevos switches virtuales compatibles con el estándar SDN para la gestión de las nuevas ONTs de nivel 2 desplegadas en la maqueta GPON.
5. Análisis de las diferentes funcionalidades que se encuentran en el programa de gestión global SDN desarrollado en Python.
6. Automatización del programa de gestión global mediante el desarrollo de las funciones necesarias en Python.
7. Análisis detallado del algoritmo de configuración dinámica de servicios.

## **1.3 Fases y Métodos**

La metodología a seguir para el desarrollo de los objetivos ha conestado fundamentalmente de las fases que se explicarán a continuación.

### **1.3.1 Fase de Análisis**

La finalidad de esta fase es aprender de forma básica cómo funcionan los componentes principales de este Trabajo de Fin de Grado:

- *Análisis de la topología de red GPON del laboratorio 2L007*: estudio de los distintos componentes de red y su modo de gestión y configuración.
- *Análisis de las herramientas software utilizadas para el despliegue de SDN*: estudio de las diferentes herramientas de software implementadas sobre el escenario de red SDN-GPON desplegado.
- *Análisis del programa de gestión global de la red SDN-GPON*: estudio del programa desarrollado en Python que gestiona la maqueta GPON bajo el estándar SDN OpenFlow.

### **1.3.2 Fase de Implementación**

Fase en la que se instalarán los nuevos dispositivos disponibles en el laboratorio, llevando a cabo las modificaciones necesarias en los dispositivos ya existentes para la actualización del escenario SDN sobre la maqueta de red GPON desplegada.

También se desarrollarán las funciones necesarias en Python para el proceso de automatización y configuración de la red GPON mediante SDN, integrándolas en el programa de gestión global desarrollado en trabajos de fin de grado anteriores a este.

### **1.3.3 Fase de Pruebas**

Última fase en la que se probará la nueva infraestructura de la red con las modificaciones realizadas y posteriormente con los nuevos dispositivos instalados con el objetivo de testear su rendimiento y las nuevas funciones de automatización desarrolladas. Para ello, se utilizarán las herramientas oportunas, que nos van permitir comprobar que los resultados obtenidos son los esperados.

### **1.3.4 Fase de Realización de los Informes**

En esta fase, se procedió a realizar los informes del TFG, en concreto, la memoria del Trabajo de Fin de Grado.

## **1.4 Estructura de la Memoria del TFG**

El Capítulo 2 describe los fundamentos de las redes PON y de SDN y las principales herramientas de software utilizadas en este trabajo acabando con la descripción de la metodología a seguir.

El Capítulo 3 contiene la descripción de la solución SDN inicial desplegada sobre la red GPON, continuando con un análisis de las prestaciones de los ordenadores usados para dicha gestión, para posteriormente describir la solución SDN final. Una vez hecho esto, se probará un nuevo puerto del OLT y las dos nuevas ONTs de nivel 2 disponibles en laboratorio, describiendo por último el proceso de despliegue de nuevos switches virtuales SDN para controlar dichas ONTs de nivel 2.

El Capítulo 4 describirá el funcionamiento de las principales funcionalidades del programa de gestión global del entorno de red SDN-GPON implementado, describiendo posteriormente, las funciones desarrolladas para la automatización de dicho programa y terminando con un análisis en profundidad del comportamiento del algoritmo de configuración dinámica de servicios.

El Capítulo 5 incluye las conclusiones finales de este Trabajo de Fin de Grado e introduce posibles líneas futuras a desarrollar a partir de este proyecto.

Al final de la memoria se incluyen la bibliografía y un Anexo donde se encuentra el código de las funciones descritas a lo largo de la memoria.

# 2

## Metodología y herramientas de trabajo

### 2.1 Introducción

En este punto de la memoria, se va a hacer, en primer lugar, una descripción de los principales fundamentos teóricos de las redes PON y de la tecnología SDN, para posteriormente analizar las herramientas software que se van a utilizar a lo largo del desarrollo del trabajo, así como el lenguaje de programación Python, usado para el desarrollo del código necesario para la gestión de la red GPON mediante tecnologías SDN. Además, se describirá la metodología a seguir para la consecución de los objetivos propuestos.

La red de acceso GPON usada para el desarrollo de este Trabajo de Fin de Grado, a la que nos referiremos en lo posterior, se encuentra en un armario en el laboratorio de Comunicaciones Ópticas de la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad de Valladolid. En dicho armario, se encuentra el OLT, las diferentes ONTs, dos divisores ópticos 1:8 y 25 kilómetros de fibra óptica separados en varias bobinas que conectan todo.

### 2.2 Redes de Acceso Ópticas Pasivas (PON, *Passive Optical Networks*)

Una red PON (*Passive Optical Network*) está formada por una serie de elementos pasivos unidos mediante fibra óptica. Dichos elementos pasivos ópticos, tales como los divisores o *splitters*, reciben este nombre al no necesitar de alimentación para su funcionamiento. Las redes PON presentan una topología en árbol con una configuración

punto multipunto para transmitir datos desde hasta los usuarios finales. A continuación, se describen los principales componentes que forman una red PON:

- OLT (*Optical Line Terminal*): elemento activo que se encuentra en la oficina central del proveedor de servicios. Se trata del punto de partida de la red de acceso y su función es convertir, entramar y transmitir señales por la red PON.
- Splitters o divisores ópticos: elementos pasivos que dividen la señal que proviene del OLT en diferentes ramas hacia los usuarios finales. Del mismo modo, estos divisores pasivos combinan las señales en el sentido inverso, esto es, desde los usuarios hacia fuera de la red PON.
- ONT (*Optical Network Termination*)/ONU (*Optical Network Unit*): se trata de un elemento activo que se encuentra ubicado en el domicilio del cliente. Convierte las señales ópticas transmitidas a través de la fibra en señales eléctricas (y viceversa) y permite enviar datos en sentido ascendente desde los usuarios hacia el OLT.

La transmisión de señales desde el OLT hacia las ONTs/ONUs, es decir en el sentido descendente, se realiza utilizando la longitud de onda de 1490 nm, mientras que en el sentido ascendente (desde las ONTs/ONUs hacia el OLT) se utiliza la longitud de onda de 1310 nm. Además, es posible separar la transmisión de video utilizando la longitud de onda dedicada de 1550 nm.

Tanto en el sentido descendente como en el ascendente, se utiliza TDMA (*Time Division Multiplexing*) como protocolo de control de acceso al medio. En el sentido descendente, el OLT, envía la información a cada ONT/ONU en modo broadcast y cada ONT/ONU accede a la información que le corresponde y omite lo demás. Por este motivo, es necesario garantizar la seguridad puesto que todos los usuarios reciben toda la información, para lo que se utilizan protocolos de encriptamiento robustos. En el sentido ascendente, TDMA separa la información de cada ONT en intervalos de tiempo diferentes sobre el mismo canal (longitud de onda) para evitar colisiones de datos de diferentes usuarios. Además, para evitar dichas colisiones es necesario que el OLT conozca perfectamente los retardos de las diferentes ONTs [1], ya que cada una estará a una distancia diferente de la OLT, esto es, de la Oficina Central (CO, *Central Office*).

## 2.3 Introducción a SDN (Software Defined Networking)

Las Redes Definidas por Software o SDN (*Software Defined Networking*) son una tecnología en la que se separa el plano de datos del plano de control, por lo que el control se desprende del hardware y se le otorga a una aplicación software, que se denomina controlador y que mantiene una visión global de toda la red. La separación del plano de control proporciona a los operadores de red la posibilidad de configurar de forma centralizada la red y sus servicios, modificándolos en tiempo real y pudiendo desplegar nuevas aplicaciones en un tiempo muy reducido comparándolo con las redes actuales convencionales [2].

Además de los controladores, los otros elementos fundamentales de una red SDN son las interfaces *Southbound* y *Northbound*. La interfaz *Southbound* permite conectar al controlador con la capa de red inferior, de modo que le permite realizar cambios en tiempo real en función de las necesidades y demandas. Por otro lado, la interfaz *Northbound* facilita al controlador la conexión con las aplicaciones de la capa superior, siendo uno de los puntos más críticos de la red, pues soportan gran variedad de servicios y aplicaciones. Por otra parte, también se definen las interfaces *East/West*, que permiten la interconexión entre controladores que pertenecen a diferentes redes físicas, de modo que un controlador SDN de un entorno determinado puede acceder a los recursos de red y a las funciones de otro controlador [3]. Todo lo explicado anteriormente queda ilustrado en la Figura 1.

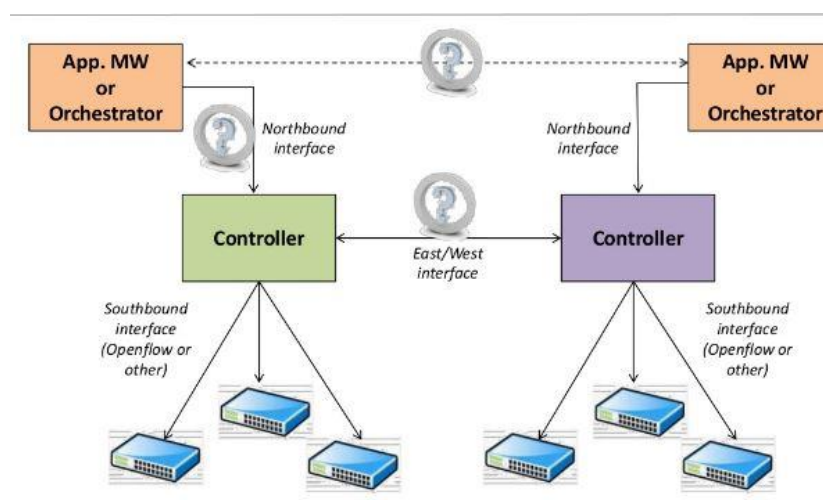


Figura 1: Ejemplo de topología de redes SDN

Por las características que se han mencionado anteriormente, SDN permite una mayor flexibilidad y agilidad, aumentando la optimización de los recursos y simplificando las redes y sus dispositivos.

## **2.4 Herramientas y software SDN utilizados**

### **2.4.1 Protocolo OpenFlow 1.3**

El estándar que se utiliza en la red GPON del laboratorio para la implementación de SDN es OpenFlow 1.3 [4]. Como ya se ha comentado anteriormente la gestión de la red se lleva a cabo a través de un controlador SDN. Dicho controlador, se comunica con los *switches* a través del envío de mensajes Openflow y es el encargado de tomar las decisiones de encaminamiento. Para ello el controlador, configura en los *switches* tablas Openflow que contienen *flows* o flujos de datos, que hacen que el switch ejecute diferentes operaciones (*instructions*) en caso de que se cumplan unas determinadas condiciones denominadas *match*.

Además, también es posible hacer que el switch tenga en cuenta diferentes parámetros para procesar los mensajes que le llegan, estableciendo diferentes prioridades para los flujos y anidando tablas.

Para el caso de uso que nos ocupa, el controlador instalado para gestionar la red del laboratorio es *OpenDayLight* (ODL) [5], cuyas características se explicarán en las siguientes secciones.

### **2.4.2 Integración de switches virtuales OVS (Open Virtual Switch)**

Para la implementación de los switches virtuales la elección es OVS (*Open Virtual Switch*) [6], un software de código abierto que soporta y es soportado por Openflow y se encarga de la creación de puentes o *bridges*. OVS nos permite la creación de una nueva capa en los dispositivos donde se implementa, transparente al usuario y al resto de la red.

El OVS, se puede separar en dos en dos componentes principales, el espacio de usuario (*ovs-vswitchd*) y el módulo *kernel* (*datapath kernel module*). Los paquetes llegan

a través del espacio de *kernel* y es el espacio de usuario quien decide que acciones se realizan sobre ellos.

En el espacio *kernel*, el OVS, está diseñado específicamente para el entorno en el que estamos trabajando, mientras que en el espacio de usuario el OVS funciona con cualquier entorno. En este último caso, los paquetes que llegan al *kernel* se envían al espacio de usuario donde se realizan los cambios correspondientes; mientras que en el primer caso es en el espacio de *kernel* donde se llevan a cabo las instrucciones pertinentes, siendo el espacio de usuario donde se toman las decisiones sobre los datos. De este modo, se aumenta en gran medida la velocidad de ejecución, eliminando el posible cuello de botella que pudiese crear el espacio de usuario, optimizando de forma considerable el ancho de banda del OVS.

En la red GPON del laboratorio, los OVS se encuentran instalados a la entrada/salida del OLT y las ONTs, los elementos activos, de forma que podamos controlar los paquetes que entran y salen de dichos elementos. Es decir, creamos una capa de abstracción a nivel software sobre el OLT y las ONTs que emulen las capas SDN sobre dichos dispositivos activos. Así pues, el OVS que controla el flujo descendente (desde la OLT hasta las ONTs) se encuentra instalado en el ordenador central (justo antes del OLT), mientras que el OVS que controla el flujo ascendente de los usuarios (desde las ONTs hasta el OLT) se encuentra instalado dentro máquinas virtuales en ordenadores justo después de las ONTs hacia los dispositivos de la red residencial del usuario. El uso de estos OVS nos permite configurar flows capaces de controlar las tasas máximas de transmisión utilizando los denominados meters, que se encargan de imponer dichas tasas de transmisión máximas.

### **2.4.3 Controlador SDN OpenDayLight (ODL)**

Como ya se ha mencionado anteriormente, en las redes SDN, una de las piezas clave es el controlador. En la red GPON desplegada en el laboratorio el controlador seleccionado es *OpenDayLight* (ODL). Se trata de un software de código abierto, programado en Java, que presenta actualizaciones constantes. Los dispositivos se presentan como objetos y permiten la instalación de únicamente los protocolos y servicios que sean necesarios para satisfacer nuestras necesidades.



Contiene una interfaz web muy completa, como se puede ver en la Figura 2, que permite la configuración de multitud de opciones y servicios. Por otro lado, existe la posibilidad de llevar a cabo la configuración de opciones a través de peticiones HTTP (*Hypertext Transfer Protocol*), usando los métodos GET, PUT y DELETE.

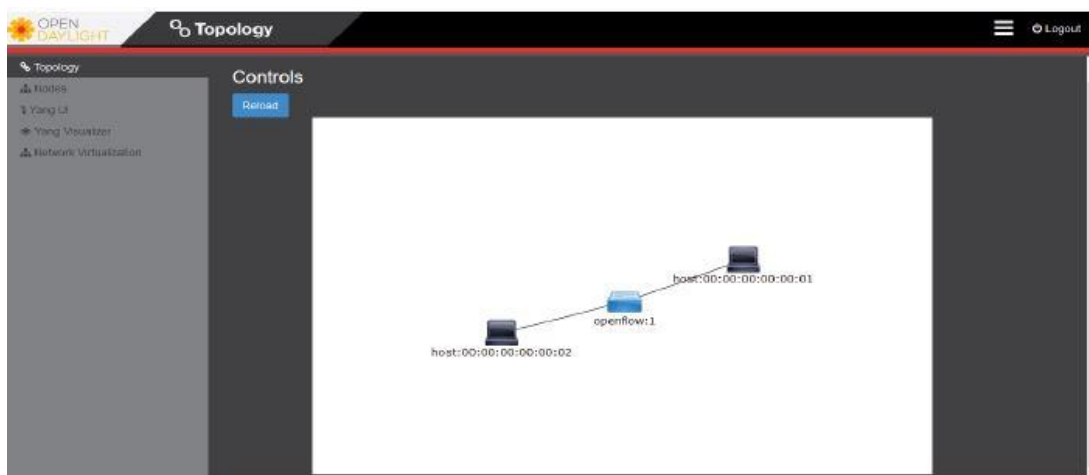


Figura 2: Interfaz web del controlador ODL

## 2.5 Lenguaje de programación Python

El código desarrollado para la gestión de servicios en la red GPON del laboratorio está implementado en el lenguaje de programación Python, por lo que a lo largo de este TFG se implementarán mejoras en dicho código para la optimización y automatización de ciertas funcionalidades SDN integradas en la red GPON.

Python es un lenguaje de programación de alto nivel, interpretado, es decir, no necesita una compilación previa del código completo antes de la ejecución, sino que a medida que se ejecuta, se va compilando línea a línea. Es multiparadigma, ya que soporta orientación a objetos, programación imperativa y programación funcional. Tiene una sintaxis muy simple y cuenta con un gran número de librerías apropiadas para la gestión de redes como la que se encuentra en el laboratorio [7].

## 2.6 Metodología de trabajo

En esta sección se va a describir la metodología y pasos seguidos para la consecución de los objetivos propuestos para este trabajo.

### **2.6.1 Integración y actualización de nuevos dispositivos PON**

Buscando el objetivo de acercar lo máximo posible la red del laboratorio a una red real, se va a llevar a cabo la integración de dos nuevas ONTs de nivel 2, semejantes a la que ya estaba integrada en la red. Para llevar a cabo su control mediante SDN, será necesaria la instalación de nuevos OVSS que permitan a las ONTs trabajar simultáneamente bajo dicho paradigma de red. Dichos OVSS, serán instalados en ordenadores nuevos, con mejores prestaciones.

Por otro lado, se va a activar un nuevo puerto PON del OLT, realizando las configuraciones necesarias para poder testear su funcionamiento posteriormente y poder compararlo con el puerto usado hasta este momento.

Además, para la lograr la máxima eficiencia posible, se pretende comprobar el funcionamiento de todos los dispositivos instalados previamente, realizando los cambios y actualizaciones necesarias para proporcionar un aprovechamiento más óptimo de los recursos disponibles en la red GPON del laboratorio.

### **2.6.2 Integración y mejora de políticas SDN y automatización de la red**

En trabajos previos se ha implementado en la red GPON del laboratorio, una solución SDN soportada por el protocolo OpenFlow 1.3, donde se utilizan switches virtuales para gestionar el tráfico de subida y de bajada en la red, controlados dichos canales por un controlador OpenDayLight instalado en una máquina virtual en el ordenador central.

Sobre este escenario, se han implementado políticas para la gestión en tiempo real de los recursos de la red, de modo que los abonados puedan contratar por ejemplo en tiempo real nuevos requisitos en los servicios contratados, tales como aumentar su ancho de banda temporalmente. Además, también se ha implementado un algoritmo para la gestión dinámica del ancho de banda de los usuarios, de modo que en caso de que un usuario necesite un mayor ancho de banda, se le asigne de forma transparente, en función de los recursos que la red tenga disponibles en ese momento. De este modo, se proponen nuevos modelos de negocio bajo demanda que podrían gozar de gran interés para los operadores de red y/o proveedores de servicio.

En este sentido, en este trabajo se pretende mejorar en la medida de lo posible las políticas de gestión y configuración ya implementadas, a través de un análisis del código desarrollado en Python, realizando los cambios oportunos que consigan un funcionamiento más óptimo de dichas políticas.

Finalmente, también se pretende dotar a la red de una mayor escalabilidad, automatizando la instalación de nuevos dispositivos que faciliten la ampliación de la red sin necesidad de realizar cambios significativos en el código que tenemos implementado. Para ello se van a desarrollar los métodos necesarios y se modificarán otros ya existentes.

## **2.7 Conclusiones**

En este capítulo de la memoria, se ha presentado el entorno SDN que se encuentra desplegado sobre un testbed GPON desplegado en un laboratorio, realizando una descripción del software y las herramientas que se van a utilizar para el desarrollo de los objetivos del trabajo. También se ha presentado el lenguaje de programación con el que se va a trabajar y la metodología a seguir para la consecución de los objetivos propuestos, comenzando con la integración de nuevos dispositivos y la actualización de los ya existentes y siguiendo con la mejora y optimización de las políticas SDN implementadas y la automatización de la red de cara a futuras ampliaciones.

# 3

## Actualización del entorno SDN-GPON

### 3.1 Introducción

En este capítulo, se va a realizar un análisis experimental de ciertos equipos que permiten el funcionamiento de la red para su posterior actualización y de este modo llevar a cabo los cambios que sean necesarios para un mejor funcionamiento de la red GPON. Además, se conectarán a la red dos nuevas ONTs de nivel 2, describiendo los pasos a seguir para su integración y configuración para finalmente comprobar su correcto funcionamiento.

Una vez hecho esto, comenzará el proceso de automatización de la red, mediante pequeñas modificaciones de algunas funciones y añadiendo otras en el código desplegado en Python que rige el funcionamiento del controlador ODL. Con esto se pretende dotar a la red de una mayor escalabilidad y flexibilidad sin necesidades de futuras modificaciones en el código.

### 3.2 Descripción del entorno SDN-GPON inicial

A continuación, se va a describir la solución SDN desplegada en el laboratorio al comienzo de este trabajo. Para ello, se va a presentar la estructura y se van a describir los componentes y los diferentes modos de gestión con los que cuenta la red de GPON desplegada en el laboratorio.

El OLT de la red GPON del laboratorio da servicio a varias ONTs que están conectadas en una topología en árbol. El ordenador central, es la puerta de enlace del laboratorio con la red externa de la facultad. Tiene conectado un cable Ethernet, que le conecta con el exterior y le proporciona conexión a Internet. Dicho ordenador, actúa en

forma de router, para lo que tiene un archivo de configuración ejecutable, donde se encuentran las instrucciones necesarias para tal efecto, y además se definen dos VLANs, la 806 y la 833. Avanzando hacia el interior de la red de acceso, nos encontramos con el OLT, que se conecta con el ordenador central para dotar al OLT de la conexión a Internet pertinente. El otro cable que entra al OLT se encuentra en una subred diferente, la 172.26.128.0/24, y se utiliza para la gestión y configuración del OLT, dando acceso al interfaz TGMS (*TELNET GPON Management System*), que es un software que se encuentra instalado en una máquina virtual instalada en el ordenador central y que se basa en una interfaz web suministrada por el fabricante de los dispositivos que permite la configuración de servicios para los clientes.

El OLT tiene cuatro puertos de salida, con una capacidad máxima de 64 ONTs por puerto, de modo que se pueden conectar hasta 256 ONTs simultáneamente, dando servicio a otros tantos clientes. En el caso que nos ocupa, el puerto que está en servicio es el 0, con 4 ONTs conectadas, siendo una de ellas de nivel 2 y las otras tres restantes de nivel 3. De este puerto sale un cable de fibra óptica de 20 kilómetros de longitud total, divididos en una bobina de 10 kilómetros y otra de 5 kilómetros, que conecta con un *splitter* 1:8, de los dos que se encuentran disponibles, donde se divide la señal para dar servicio a las ONTs conectadas a través de una bobina de una longitud total de 5 kilómetros, compuesta por 48 fibras ópticas. Finalmente, encontramos las cuatro ONTs que ya se han comentado anteriormente. En la Figura 3, se puede ver el aspecto real de la red GPON desplegada.

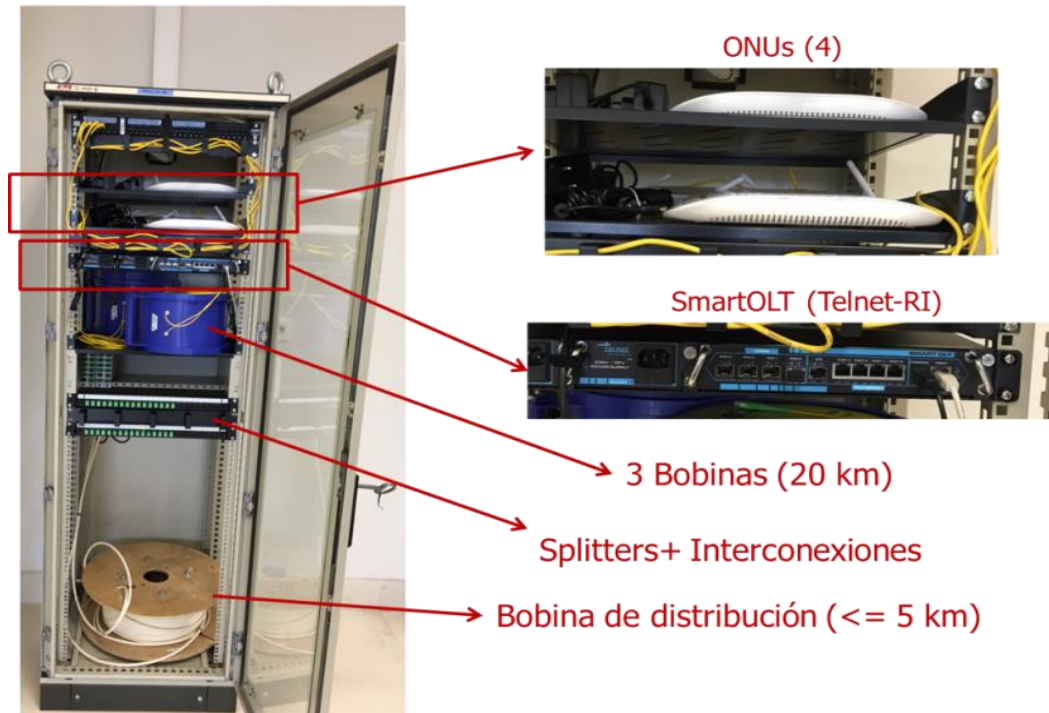


Figura 3: Aspecto real de la red de acceso GPON desplegada en el laboratorio

La interoperabilidad entre los diferentes dispositivos está garantizada, ya que todos ellos han sido fabricados por la empresa TELNET Redes Inteligentes [8]. A continuación, se van a describir cada uno de los equipos de la red GPON:

- **SmartOLT-350:** equipo utilizado en redes GPON que cuenta con 4 puertos que dan servicio a un total de 256 ONTs (64 por puerto). También cuenta con un puerto *Gigabit Ethernet* asociado a cada puerto PON, además de un puerto *FastEthernet* que admitirá tasas de hasta 10 Gbps. Para su gestión se utiliza la interfaz web TGMS proporcionada por el fabricante y utiliza el protocolo OMCI (*ONU Management and Control Interface*) para la gestión remota de las ONTs. Además, para poder configurar servicios de una forma más completa y a más bajo nivel, podemos conectarnos a través de CLI, pudiendo configurar la red GPON manualmente utilizando comandos propios del estándar.
- **ONT WaveAccess 3021:** dispositivo de nivel 3, de alto rendimiento, con router integrado en la propia ONT, que cuenta con 3 puertos *Gigabit Ethernet*, un puerto POTS destinado a la telefonía y servicio de Wifi b/g/n.

- **ONT WaveAccess 4022:** equipo con características similares al anterior, aunque mejora sensiblemente sus prestaciones, ya que cuenta con 4 puertos *Gigabit Ethernet* por los tres que presentaba la ONT WaveAccess 3021 y 2 puertos POTS en lugar de uno.
- **ONT WaveAccess 512:** ONT, a diferencia de las anteriores, de nivel 2, lo que quiere decir que no cuenta con capacidad de enrutamiento, por lo que necesita de un *router* o un *switch* a su salida que efectúe dicho trabajo. Fácilmente adaptable a cuadros eléctricos u oficinas, ya que presenta formato en carril DIN.

Los equipos que se acaban de describir trabajan en dos canales o flujos, en función de la dirección de los datos que se transmiten a través de la red. En concreto:

- **Canal descendente:** denominado *downstream*, se trata de una comunicación punto multipunto, ya que los datos viajan desde OLT situado en la oficina central del proveedor de servicios hacia las ONTs ubicadas en las viviendas de los clientes. Atendiendo a las especificaciones del estándar GPON utiliza la longitud de onda de 1490 nm y permite una tasa máxima de 2.5 Gbps
- **Canal ascendente:** denominado *upstream*, canal en el que los datos viajan desde las diferentes ONTs hasta el OLT, por lo que se trata de una comunicación punto punto. En este caso, la longitud de onda marcada por el estándar GPON es la de 1310 nm y la tasa máxima para este flujo es de 1.25 Gbps.

Un dato siempre a tener en cuenta para la configuración de servicios, es la tasa máxima de datos por flujo, ya que la suma de los servicios de todas las ONTs no puede superar en ningún momento la tasa máxima marcada por el estándar GPON en cada caso. En el caso de que esto suceda la red mostrará un error de configuración.

Sobre el testbed GPON descrito anteriormente se ha desplegado una solución SDN, implementada mediante un controlador externo y varios switches. Para poder controlar el OLT y las ONTs utilizando Openflow se ha implementado una capa de abstracción de hardware SDN. La solución propuesta contempla la utilización de switches virtuales Openflow (OVS, *Open Virtual Switch*) y un controlador OpenDayLight (ODL), aunque para el caso del controlador existen otras opciones igualmente válidas como ONOS [9].

De este modo, el controlador SDN, gestionado por el operador de red, es capaz de modificar dinámicamente y en tiempo real servicios, adecuándolos a la capacidad de la red GPON, trabajando con el tráfico en ambos canales, tanto ascendente como descendente, y atendiendo a los diferentes requisitos de QoS contratados por los clientes, como puede ser un ancho de banda garantizado asociado a los clientes dados de alta en la red GPON.

Dado que OLTs y ONTs basados en SDN aún no están disponibles, no es posible integrar los OVSs dentro de dichos dispositivos, por lo que es necesario instalarlos en dispositivos externos como puedan ser dispositivos tales como Raspberry Pi, Banana Pi, mini ordenadores u ordenadores. Particularizando sobre nuestro escenario, el OVS central (COVS), que controla todo el tráfico descendente, se encuentra instalado en un ordenador ubicado junto antes del OLT, de manera que emula la capa SDN del OLT. Por su parte, en el lado de los usuarios, justo a la entrada de cada ONT se integra un OVS remoto (ROVS), capaz de controlar el tráfico en el sentido ascendente que sale de la red residencial del usuario. Dichos dispositivos necesitan una capacidad de cálculo mucho menor, por lo que es posible utilizar dispositivos más simples como Raspberri Pi, mini ordenadores u otras placas de características similares. La arquitectura SDN-GPON descrita se puede ver en la Figura 4.

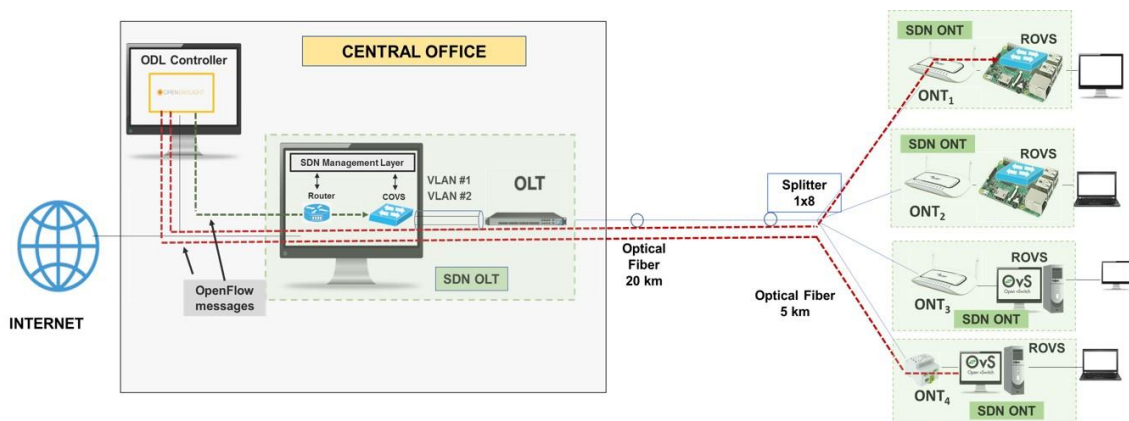


Figura 4: Esquema del escenario SDN-GPON desplegado sobre la red GPON tradicional

De este modo, el controlador ODL establece una configuración SDN a través del envío de mensajes OpenFlow a los diferentes ROVS y al COVS. En primer lugar, el controlador envía las tablas OpenFlow a los OVSs con una configuración inicial, para posteriormente poder modificar sus entradas en tiempo real en función de los servicios demandados por los usuarios o cambios en sus requisitos de QoS, por lo que por cada



servicio demandado se crearán dos flows, uno para el tráfico en el sentido descendente y otro para el tráfico del canal ascendente.

Dentro del escenario de red SDN-GPON desplegado se puede controlar por ejemplo el ancho de banda máximo, y para ello la tasa de ancho de banda asignado a cada servicio y usuario se mide a través de meters OpenFlow. Dichos meters se asocian directamente con los flows y permiten controlar la tasa máxima de los paquetes que tienen asignados dichos flows. Cada meter consta de un identificador y una banda, que especifica la velocidad asociada al flow y la forma de procesar los paquetes del flow. La velocidad de los datos es medida continuamente, de modo que si dicha tasa está por encima de la establecida para el flow, se descartan paquetes, controlando, de este modo el ancho de banda máximo asociado a dicho flow. Cabe destacar, que se asume que los usuarios finales no conocen estos conceptos de bajo nivel, por lo que, este proceso se resume en comandos de nivel superior entendibles para los usuarios.

Como ya se ha comentado con anterioridad la red GPON en el lado del cliente dispone de ONTs de nivel 2 y de nivel 3. Para la solución SDN es preferible el uso de las ONTs de nivel 2, ya que nos proporcionan una mayor flexibilidad al no contar con routers propietarios y nos permite desarrollar capacidades de enrutamiento adaptadas a las necesidades y requisitos que se deseen desplegar. De este modo, cada ROVS en el lado de las ONTs necesita un servidor DHCP asociado que proporcione direccionamiento IP a los dispositivos conectados a la ONT correspondiente. De este modo, un servidor DHCP global, controlado por el operador asigna un rango de direcciones a cada servidor DHCP local, para que este pueda asignar una dirección IP a cada dispositivo conectado. De esta manera, cada dispositivo conectado a la red tendrá una dirección IP única.

### **3.3 Análisis del rendimiento de las tarjetas de red de los ordenadores que componen el escenario SDN-GPON**

#### ***3.3.1 Diseño del escenario de red para analizar el rendimiento***

En este apartado de la memoria se va a analizar el comportamiento de las tarjetas de red en términos de rendimiento (throughput) de los diferentes ordenadores que componen el escenario SDN de la red GPON. Así pues, para realizar el análisis del

throughput de las diferentes tarjetas de un ordenador, hemos realizado el proceso en el que se conectan dos ordenadores de forma directa a través de dos tarjetas de red de dichos ordenadores. Un ejemplo de la arquitectura que queremos analizar se muestra en la Figura 5.

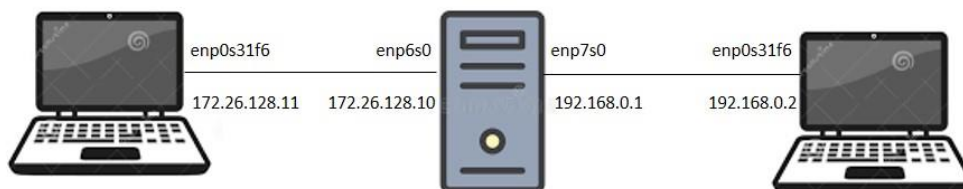


Figura 5: Arquitectura para comprobar el rendimiento de las tarjetas de red de un ordenador

Para conseguir dicha configuración los pasos a seguir son los siguientes:

1. Conectamos un cable de red entre la tarjeta de red de uno de los portátiles (Lenovo) y una de las tarjetas de red del ordenador que queremos probar y otro cable de red entre la otra tarjeta que queremos probar del mismo ordenador y el otro portátil (Lenovo).
2. Dado que no tenemos ningún servidor que nos proporcione direccionamiento IP, debemos dar direcciones IP fijas a todas las interfaces que vamos a usar, tanto en el ordenador de sobremesa que queremos probar como en los ordenadores portátiles (extremos de la conexión). Para ello hacemos clic en *Sistema* → *Preferencias* → *Conexiones de red*. Se abre una ventana y seleccionamos la interfaz que queremos editar. Pinchamos en la pestaña “*Ajustes de IPv4*” y en el desplegable seleccionamos “*Método Manual*”. A continuación, pulsamos el botón de *Añadir*. Rellenamos los parámetros *Dirección* (ej: 172.26.128.11), en función de la interfaz que estemos configurando de las que se pueden ver en la Figura 5 y *Máscara de Red* con “24”. Hay que tener en cuenta que las direcciones de las interfaces que están conectadas mediante el cable deben pertenecer a la misma subred. El resto de parámetros pueden quedar en blanco. A continuación, presionamos en *Guardar* y ya tendríamos la dirección IP que hayamos indicado asignada a la interfaz. Para comprobar que esto es así, abrimos una terminal y tecleamos el comando *ifconfig*. En caso de que la interfaz que estamos configurando ya tuviese una dirección asignada manualmente, tenemos que

reiniciar el ordenador para que los cambios tengan efecto. En la Figura 6, podemos ver cómo quedaría la asignación del direccionamiento IP.

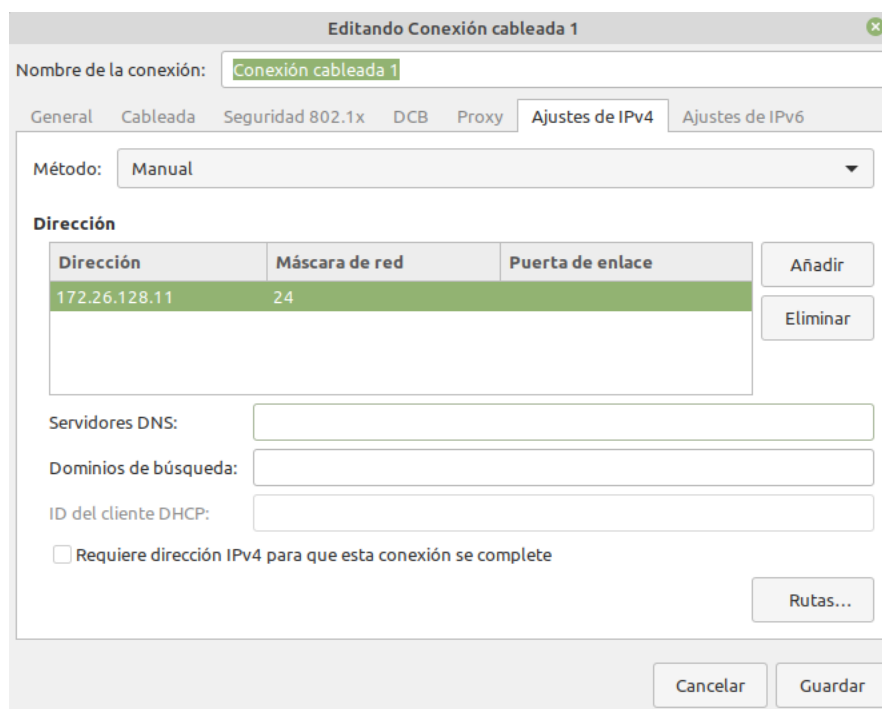


Figura 6: Asignación estática de direcciones IP en los ordenadores

Alternativamente, también podemos hacer este proceso desde terminal. Para ello, en primer lugar, nos identificamos como superusuarios tecleando `sudo su -`, introducimos la contraseña y tecleamos el comando `ifconfig` seguido del nombre de la interfaz y la dirección IP que queremos asignar junto con la máscara de red. Un ejemplo de ello sería así: `ifconfig enp7s0 192.168.0.1/24`.

3. Por otro lado, tenemos que hacer que el ordenador central (del que queremos probar sus tarjetas) funcione como un router para que dirija los paquetes de una de las subredes que hemos definido a la otra, es decir, para que pase los paquetes de una tarjeta de red a la otra a la que están conectados los ordenadores portátiles. Para ello no tenemos más que escribir un "1" en el fichero `ip_forward` que se encuentra en la ruta `/proc/sys/net/ipv4/ip_forward`. Así pues, abrimos una terminal y nos identificamos como superusuarios tecleando `sudo su -` y nos pedirá la contraseña. A continuación, nos vamos a la ruta del archivo y escribimos el "1" del siguiente modo `echo 1 > /proc/sys/net/ipv4/ip_forward`.

4. Del mismo modo, en los ordenadores portátiles tenemos que indicar por donde tienen que ir los paquetes que van dirigidos hacia la otra subred que hemos definido. Para ello, tenemos que establecer una ruta estática indicando la subred que queremos alcanzar y a través de qué dirección conocida vamos a llegar a ella. Así pues, abrimos una terminal, nos identificamos como superusuarios y utilizamos el comando `ip route add`. De forma más concreta haremos:

- Para el portátil de la izquierda en la Figura 5 el comando sería `ip route add 192.168.0.0/24 via 172.26.128.10`.
- Análogamente, en el portátil de la derecha (Figura 5) sería `ip route add 172.26.128.0/24 via 192.168.0.1`.

Para comprobar que se ha definido la ruta correctamente podemos consultar la tabla de enrutamiento del equipo tecleando `route -n`. Podemos ver un ejemplo de la salida de dicho comando en la Figura 7.

```
tfglab7@tfglab7-B150M-D3H ~ $ route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 10.0.103.253 0.0.0.0 UG 0 0 0 enp0s31f6.833
10.0.103.0 0.0.0.0 255.255.255.0 U 0 0 0 enp0s31f6.833
10.19.59.0 0.0.0.0 255.255.255.0 U 0 0 0 br0
169.254.0.0 0.0.0.0 255.255.0.0 U 1000 0 0 enp0s31f6.833
172.26.128.0 0.0.0.0 255.255.255.0 U 0 0 0 enp4s0
192.168.0.0 0.0.0.0 255.255.255.0 U 0 0 0 br0
```

Figura 7: Ejemplo de salida del comando `route -n`

De este modo ya tenemos nuestra red configurada con las dos subredes listas para llevar a cabo la prueba. Para ello, mediante la herramienta `iperf` [10] podemos comprobar cuál es el ancho de banda máximo que soporta la red que acabamos de configurar. Podemos llevar a cabo esta prueba utilizando una conexión UDP o TCP en función de nuestras necesidades. Si queremos utilizar una conexión TCP tenemos que seguir los siguientes pasos:

1. En primer lugar, tenemos que poner uno de los dos portátiles en modo escucha (cliente) tecleando en una terminal `iperf -s`.
2. Desde el otro portátil en una terminal tecleamos `iperf -c` seguido de la dirección IP del portátil que tenemos escuchando, a modo de servidor. Por ejemplo, en la Figura 5 si tenemos escuchando al portátil de la izquierda, en el portátil de la derecha deberíamos teclear `iperf -c 172.26.128.11`. A continuación, esperamos unos segundos y en cualquiera de los dos nos aparecerá el ancho de banda máximo que soporta la tarjeta de red.

Si por el contrario la conexión que queremos utilizar es UDP, tenemos que indicar tanto en el ordenador que utilizamos como servidor como en el que utilizamos como cliente que dicha conexión es UDP. En el servidor tecleamos `iperf -s -u` y en el cliente `iperf -c 172.26.128.11 -b 1000m -u`, siendo el parámetro que sigue a la opción `-b` lo suficientemente grande para que `iperf` nos devuelva el ancho de banda máximo que soporta la red.

En caso de que queramos visualizar los resultados obtenidos mediante una gráfica en Wireshark, tal y como se observa en la Figura 8, el proceso sería el siguiente:

1. En uno de los ordenadores, el que actuará como origen, tecleamos en una terminal el comando:

```
iperf -u -c 172.26.128.11 -b 600000000 -t 900
```

siendo los parámetros:

- u: indica que la conexión es UDP.
- c: indica la dirección IP del destino.
- b: tasa máxima de ancho de banda.
- t: tiempo que dura la conexión.

2. En el otro ordenador, en este caso el destino, es donde vamos a realizar la captura de paquetes mediante Wireshark para poder visualizar posteriormente la gráfica. Nos identificamos como superusuario y tecleamos en una terminal el comando:

```
dumpcap -i enp0s31f6 -f "ip dst 172.26.128.11" -w/root/prueba.pcapng
```

siendo los parámetros:

- i: interfaz por la que recibimos los paquetes.
- f: filtro de captura, en este caso la dirección IP de destino.
- w: ruta donde vamos a guardar el archivo con formato `.pcapng`.

Para cortar la captura de paquetes pulsamos `Ctrl+C`.

3. A continuación tecleamos el comando `wireshark prueba.pcapng`. Se abrirá Wireshark y cuando termine de cargar toda la captura pinchamos en `Statistics` → `I/O Graph` y ya podremos visualizar la gráfica.

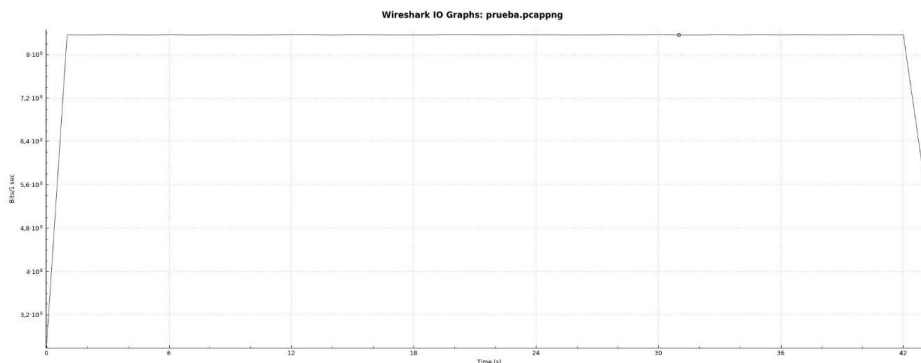


Figura 8: Gráfica ejemplo con Wireshark

### 3.3.2 Análisis del rendimiento de las tarjetas en el ordenador central del escenario SDN-GPON

En este caso el proceso es ligeramente diferente, ya que dicho ordenador tiene algunas características predefinidas como se puede ver en la Figura 9. El equipo tiene tres tarjetas de red, una integrada y dos externas. La VLAN 833 está definida por defecto y asociada a la interfaz enp4s0 con dirección IP 10.0.103.48, por lo que en el portátil que conectemos en esta interfaz debemos definir también la VLAN con etiqueta 833. Para ello, existe un archivo de configuración denominado *activarServidor.sh* en los ordenadores portátiles. Dicho archivo, contiene las instrucciones necesarias para, además de definir la VLAN 833, asignar la dirección IP correspondiente y establecer una ruta de encaminamiento apropiada. En algún caso, deberemos modificar, si es necesario, la dirección IP que queremos asignar a nuestra VLAN, así como la ruta de encaminamiento. Para ejecutarlo no tenemos más que abrir un terminal, identificarnos como superusuarios y teclear *sh activarServidor.sh*. En la Figura 10, se muestra el código de dicho archivo.

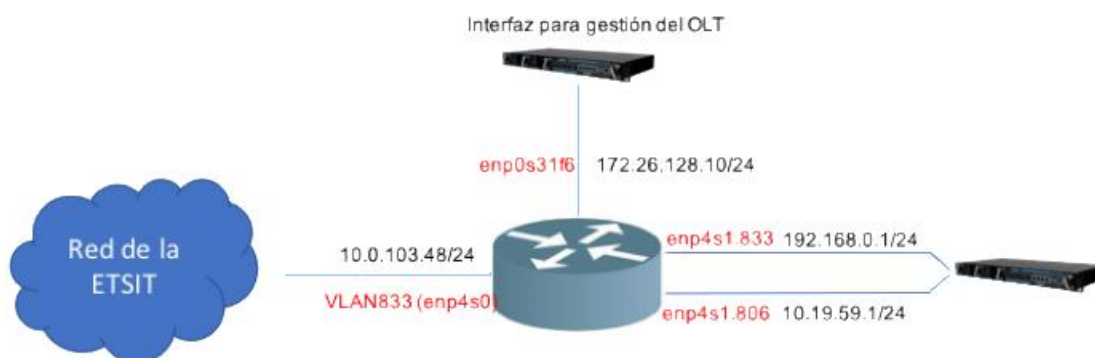


Figura 9: Topología de red del ordenador central

```

activarServidor.sh x
ifconfig enp0s31f6 0
vconfig add enp0s31f6 833
ifconfig enp0s31f6.833 0
ifconfig enp0s31f6.833 10.0.103.60/24
route add default gw 10.0.103.48 dev enp0s31f6.833
    
```

Figura 10: Código del archivo activarServidor.sh

En el ordenador central, la interfaz *enp031f6* tiene una dirección IP ya asignada, es la 172.26.128.10, por lo que la dirección que demos al portátil que conectemos en dicha interfaz debe adecuarse a esta, es decir, pertenecer a la misma subred.

Por otro lado, la interfaz *enp4s1* hay que levantarla, y para ello ejecutamos el archivo *activar-enrutamiento.sh*. Este archivo, contiene las instrucciones necesarias para, además de levantar la interfaz y definir las VLAN 833 y 806, también se incluya la instrucción *echo 1 > /proc/sys/net/ipv4/ip\_forward* que hace al ordenador funcionar como un router. De este modo, las arquitecturas que se han probado han cubierto todas las combinaciones posibles para analizar el rendimiento de las tres tarjetas de red del ordenador central, tal y como se muestra en las Figuras 11, 12 y 13.

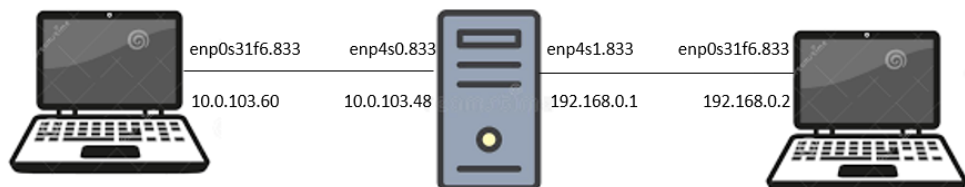


Figura 11: Arquitectura de la red para el escenario 1

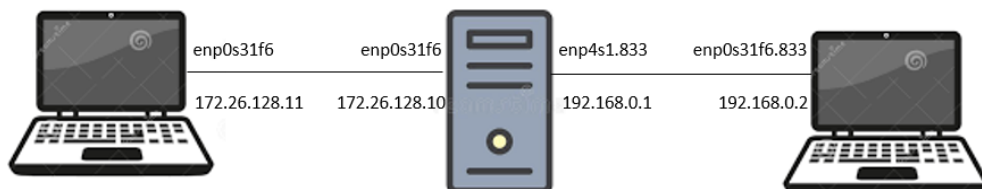


Figura 12: Arquitectura de la red para el escenario 2

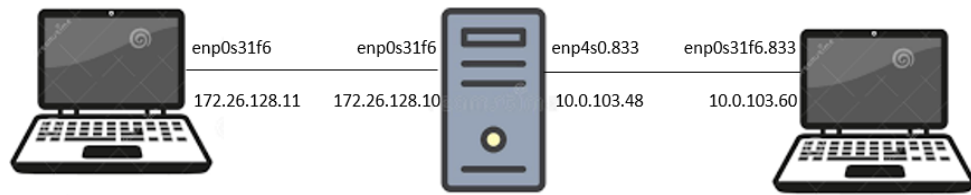


Figura 13: Arquitectura de la red para el escenario 3

Los resultados obtenidos para cada uno de los escenarios definidos se recogen en la Tabla 1:

<b>Interfaz de entrada/Interfaz de salida</b>	enp0s31f6	enp4s0.833	enp4s1.833
enp0s31f6	—	653 Mbps	664 Mbps
enp4s0.833	811 Mbps	—	287 Mbps
enp4s1.833	803 Mbps	306 Mbps	—

Tabla 1. Resultados de las pruebas realizadas del rendimiento de las tarjetas del ordenador central.

Tal y como se observa en los resultados de la Tabla 1, la interfaz enp0s31f6 corresponde con la tarjeta de red integrada, mientras que las interfaces enp4s0.833 y enp4s1.833 van asociadas a las tarjetas de red externas, cuyo modelo es Intel pro/1000 GT [11]. Además, las pruebas con interfaces de entrada enp4s0.833 y enp4s1.833 e interfaz de salida enp0s31f6 han tenido que llevarse a cabo utilizando conexiones UDP, ya que entre estas interfaces la herramienta iperf presentaba problemas en el modo TCP. En caso de que queramos trabajar sin VLANs para hacer las pruebas, podemos desactivarlas tecleando *ifconfig nombre\_interfaz down*. Para volver a activarlas, el comando es el mismo solo que sustituyendo *down* por *up*. Sin embargo, los resultados que se han obtenido trabajando sin VLANs son muy semejantes a los presentados en la Tabla 1 teniendo en cuenta las VLANs.

Como se puede ver en la Tabla 1, existe un problema cuando interaccionan las interfaces enp4s0 y enp4s1, es decir, las dos tarjetas de red externas, ya que el ancho de banda transmitido en el origen no es el que llega a la otra interfaz. Se ha podido comprobar que el problema está a la salida de la tarjeta de red, ya que internamente el ancho de banda que se obtiene es máximo. Este problema puede deberse al tratamiento interno que hacen las tarjetas de red de los paquetes cuando la interacción se produce,



como es el caso, entre dos tarjetas de red externas, ya que comparten el bus PCI (*Peripheral Component Interconnect*) o incluso también pueden verse afectadas por los drivers que utilizan para su funcionamiento. Para el resto de configuraciones el resultado es adecuado, ya que se ha podido comprobar que la tasa máxima de transmisión entre las dos tarjetas de los ordenadores Lenovo cuando los conectamos directamente es de unos 800 Mbps. En concreto, podemos observar que la tasa máxima que se ha podido obtener ronda los 800 Mbps cuando la entrada es cualquiera de las dos tarjetas de red externas y la salida es la tarjeta de red integrada; sin embargo cuando la transmisión toma como entrada la tarjeta de red integrada en el equipo y salida cualquiera de las dos externas, el resultado obtenido es de alrededor de 650 Mbps, que es algo inferior al obtenido en el sentido inverso, pero aún así es muy superior a la tasa máxima obtenida entre las dos tarjetas externas.

### **3.3.3 Análisis del rendimiento de las tarjetas en los ordenadores nuevos**

En primer lugar, para poner los resultados en su contexto, se van a describir las características básicas de estos ordenadores, ya que lógicamente, de dichas características van a depender en gran medida los resultados que se mostrarán posteriormente. Estos ordenadores cuentan con un procesador modelo INTEL I7 9700 3.0Ghz 12mb socket 1151 BOX [12], una placa base modelo GIGABYTE GA-Z390 DESIGNARE [13] y dos tarjetas de red externas modelo TPLINK TG-3468 LP [14].

En este caso, no hay ninguna VLAN definida ni ninguna dirección IP asignada por defecto, por lo que el proceso es el general definido al principio de este apartado. Este ordenador cuenta con cuatro tarjetas de red, dos de ellas son integradas y las otras dos son externas. La configuración general es la que se muestra en la Figura 14.

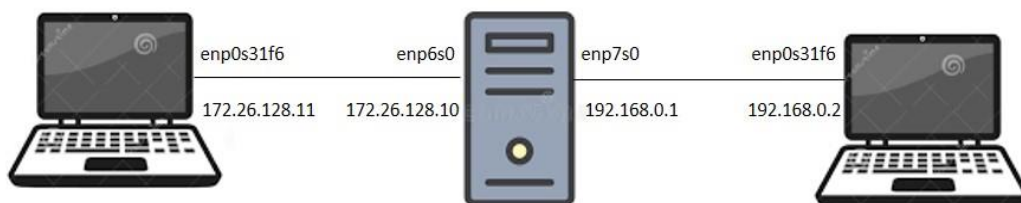


Figura 14: Arquitectura de red para analizar el rendimiento de las tarjetas del ordenador nuevo

Los resultados obtenidos de dicho análisis se recogen en la Tabla 2. Cabe destacar que las tarjetas de red integradas son las que se corresponden con las interfaces eno2 y enp3s0, mientras que las dos tarjetas externas son las que corresponden con las interfaces enp6s0 y enp7s0.

<b>Interfaz de entrada/Interfaz de salida</b>	eno2	enp3s0	enp6s0	enp7s0
eno2	—	665 Mbps	661 Mbps	667 Mbps
enp3s0	665 Mbps	—	675 Mbps	668 Mbps
enp6s0	681 Mbps	669 Mbps	—	669 Mbps
enp7s0	685 Mbps	677 Mbps	751 Mbps	—

Tabla 2. Resultados de las pruebas realizadas del rendimiento de las tarjetas del ordenador nuevo.

Tal y como se observa en los resultados, aunque los resultados son mucho más estables en todas las configuraciones que en el caso del ordenador central, son algo inferiores ya que, aunque se alcanza un pico de 751 Mbps, la tónica general muestra una tasa de alrededor de 675 Mbps; sin embargo, es una tasa máxima suficiente para el uso que se va a hacer del ordenador. Estas diferencias en los resultados con respecto al ordenador central, pueden deberse a cuestiones de hardware, o, incluso al propio programa Iperf, que, aunque es un programa aparentemente sencillo, en ocasiones su configuración interna es complicada. Todas las pruebas se han llevado a cabo utilizando tanto iperf como iperf3 obteniéndose los mismos resultados.

### **3.3.4 Configuración del nuevo escenario de red SDN-GPON**

Como se ha comentado anteriormente, se detectó que existía un problema de rendimiento (*throughput*) cuando se producía la interacción entre las interfaces que corresponden con las tarjetas de red externas del ordenador central, ya que se producía una caída muy brusca en el ancho de banda recibido. Dicho problema, tiene una influencia significativa en el funcionamiento de la red con la actual topología, ya que limita de una manera importante la capacidad de la red. Después de barajar varias opciones, la decisión aplicada, por ser la más práctica y segura, fue la de hacer un intercambio entre el papel que juegan las interfaces enp4s0 y enp0s31f6. De este modo la topología final es la que se puede ver en la Figura 15. De tal manera que una de las interfaces externas, en este caso enp4s0 se conecta al interfaz de gestión del OLT que no necesita dicha comunicación de niveles muy altos de capacidad. Mientras, la otra interfaz

externa se conectaría al interfaz del OLT (enp4s1) y la integrada a la salida de Internet (enp0s31f6). Con esta configuración, tendríamos conexiones con capacidades cercanas a los 700 Mbps, suficiente para las pruebas que se desean realizar con el escenario SDN-GPON.



Figura 15: Topología de red final en el ordenador central

Para llevar a cabo estos cambios entre las interfaces de red, tenemos que modificar algunos archivos y opciones que voy a indicar a continuación. En primer lugar, tenemos que acceder al archivo del ordenador central donde están definidas las interfaces y las VLANs. Este archivo se encuentra en la ruta `/etc/network/interfaces` y para acceder a él, tenemos que teclear `nano`, seguido de la ruta, después de habernos identificado como superusuarios. Una vez dentro del archivo, no tenemos más que establecer la VLAN 833 en la interfaz `enp0s31f6` y quitarla de la interfaz `enp4s0`. Además, en este mismo archivo, a la interfaz `enp4s0` le asignamos la dirección IP `172.26.128.10` de forma estática, así, tanto la VLAN como la dirección IP quedan establecidas de forma permanente. En la Figura 16 podemos ver como queda finalmente el archivo.

```

GNU nano 2.5.3                               File: /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

auto enp0s31f6.833
iface enp0s31f6.833 inet static
    address 10.0.103.48
    netmask 255.255.255.0
    broadcast 10.0.103.255
    network 10.0.103.0
    gateway 10.0.103.253

auto enp4s0
iface enp4s0 inet static
    address 172.26.128.10
    netmask 255.255.255.0
    broadcast 172.26.128.255
    network 172.26.128.0
    gateway 172.26.128.6

# interfaz enp4s1
iface enp4s1 inet static
    address 192.168.0.1
    netmask 255.255.255.0
    broadcast 192.168.0.255
    network 192.168.0.0

#VLAN 833
iface enp4s1.833 inet static
    address 192.168.0.1
    netmask 255.255.255.0
    broadcast 192.168.0.255
    network 192.168.0.0
    vlan_raw_device enp4s1
    
```

Figura 16: Configuración final del archivo donde se definen las interfaces y las VLANs

A continuación, tenemos que modificar el archivo *activar-enrutamiento.sh*, sustituyendo la interfaz *enp4s0* por la *enp0s31f6* para establecer las nuevas rutas de encaminamiento. El archivo queda como el mostrado en la Figura 17.

```

echo 1 > /proc/sys/net/ipv4/ip_forward

iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o enp0s31f6.833 -j MASQUERADE
#iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o enp0s31f6 -j MASQUERADE
iptables -t nat -A POSTROUTING -s 10.19.59.0/24 -o enp0s31f6.833 -j MASQUERADE
#iptables -t nat -A POSTROUTING -s 10.19.59.0/24 -o enp0s31f6 -j MASQUERADE
#iptables -t nat -A POSTROUTING -s 10.19.59.0/24 -o VLAN806 -j MASQUERADE
#iptables -t mangle -A POSTROUTING -d 192.168.0.0/24 -j CLASSIFY --set-class 0:5
    
```

Figura 17: Configuración final del archivo activar-enrutamiento

Por último, en Virtualbox, tanto en la máquina virtual de TGMS como en la del controlador, tenemos que cambiar el adaptador de red por el correspondiente. Para ello accedemos a *Configuración* → *Red* como se puede ver en la Figura 18 y la Figura 19.



Figura 18: Configuración final de red de la máquina virtual del controlador

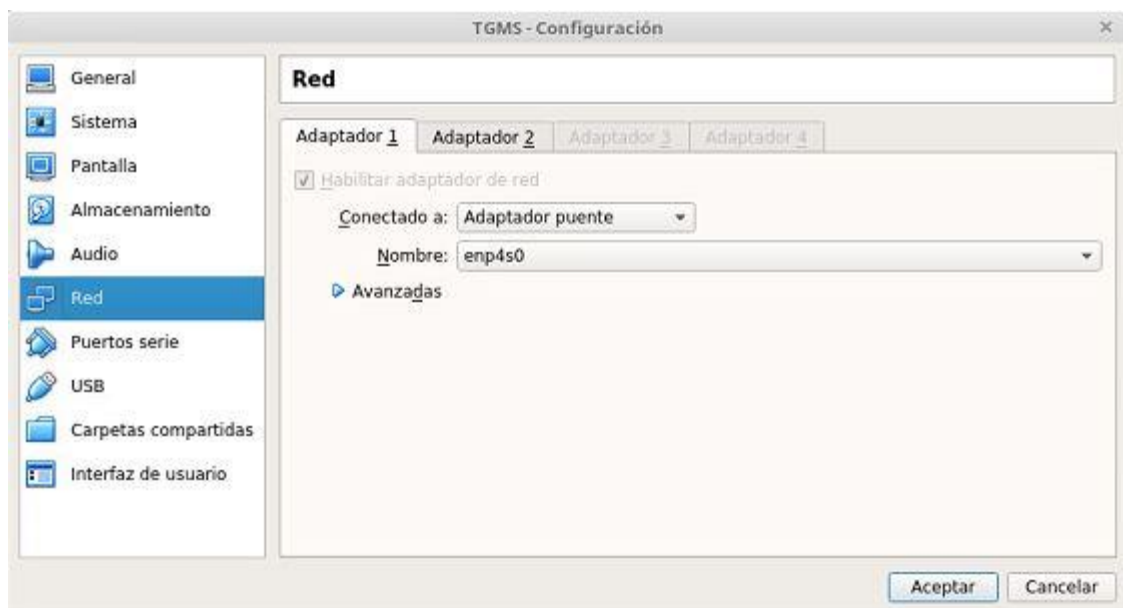


Figura 19: Configuración final de red de la máquina virtual de TGMS

Con respecto al código en Python en el que están desarrolladas las diferentes aplicaciones de la red, los cambios son mínimos. En el archivo *StatsOVS.py*, que controla el algoritmo de asignación dinámica de ancho de banda, en el punto en el que se determina si el enlace es de subida o de bajada y en el método *ShowOVS()*, donde se abren sockets que escuchan el tráfico, se ha sustituido la interfaz *enp4s0.833* por la *enp0s31f6.833* que es la que interviene en este caso.

### 3.4 Activación de nuevos puertos PON en el OLT

En este apartado de la memoria se va a describir la activación y puesta en marcha de uno de los puertos PON del OLT, en concreto el puerto PON 1 y comprobar su rendimiento. Cabe destacar que nuestro modelo de OLT SmartOLT 350 [15] viene equipado con 4 puertos PON y hasta ahora teníamos activo el puerto PON 0. Además, es importante indicar que, si tenemos conectado el puerto PON 1, el cable que conecta el OLT con el ordenador central debe estar conectado en el puerto correspondiente, en este caso sería el PORT 1. En la Figura 20 podemos ver la imagen de la parte del OLT donde se encuentran los puertos y los puertos PON.



Figura 20: Imagen frontal real del OLT y los puertos PON asociados

#### 3.4.1 Registro de las ONTs en puertos PON mediante TGMS

Una vez realizada la conectividad correcta entre puertos, el siguiente proceso es registrar las ONTs en dicho puerto PON a través del TGMS. El proceso de registro de las ONTs en TGMS es sencillo, simplemente hay que seguir los pasos que se describen a continuación:

1. En el ordenador central, abrimos una terminal y nos identificamos como superusuarios, `sudo su` – e introducimos la contraseña. A continuación, desde el terminal, abrimos virtualbox. Es importante hacerlo así, ya que, de lo contrario no aparecerá la máquina virtual asociada al TGMS.
2. Una vez dentro de virtualbox, iniciamos la máquina virtual de nombre TGMS. Tarda unos minutos en arrancar y una vez lo haga, la dejamos abierta, sin necesidad de introducir ni usuario ni contraseña.
3. Abrimos un navegador y vamos a la dirección 172.26.128.1, que seguramente esté guardada en el historial. Nos encontramos con el entorno web de TGMS y metemos el usuario y las contraseñas pertinentes.

4. A continuación, seleccionamos *PON 0*, que es el nombre dado a nuestra OLT, seguidamente *Port: 1*, que en este caso es el puerto que tenemos conectado y finalmente *Details*, como se puede ver en la Figura 21.

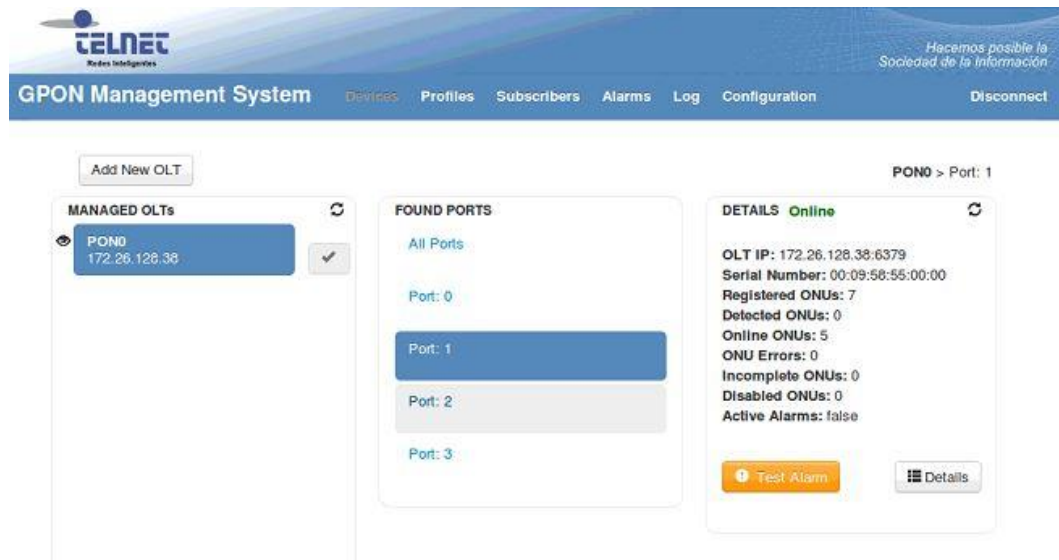


Figura 21: Acceso en TGMS a los detalles del puerto 1 del OLT

5. A continuación, se observa una ventana donde nos aparecen las ONUs que tenemos registradas en el nuevo puerto, en nuestro caso no tenemos ninguna, por lo que aparece vacía. Las ONUs que tenemos conectadas a la red GPON en todos los puertos PON, pero que todavía no están asociadas al puerto, las podemos ver en la pestaña *Disabled ONUs*, como se observa en la Figura 22.

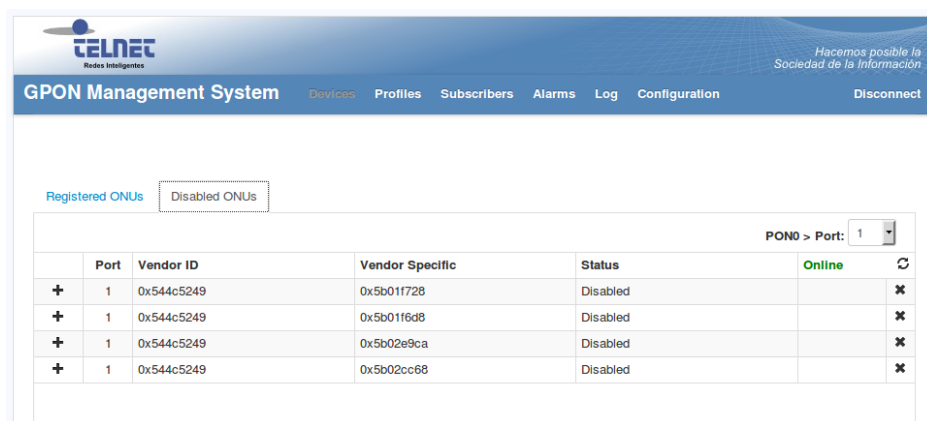
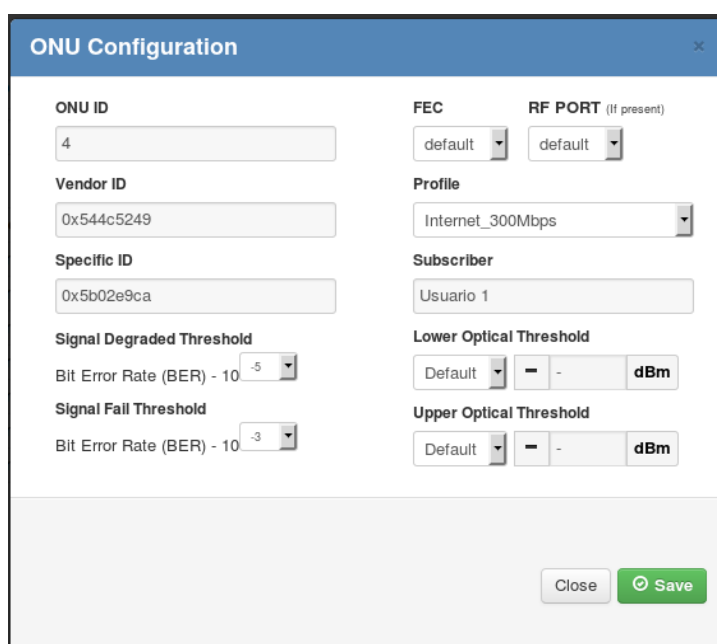


Figura 22: Pestaña con las ONUs que están conectadas en la red PON (*Disabled ONUs*) pero no asociadas al puerto Port 1

Como se observa en dicha captura se asigna el mismo *Vendor ID* para todas las ONUs que conectemos, pero el parámetro *Vendor Specific* es identificativo para cada una y corresponde con parte de su dirección MAC. Posteriormente, lo usaremos para registrar cada ONU.

6. A continuación, volvemos a la pestaña *Registered ONUs* y pinchamos en el símbolo +, que se encuentra arriba a la izquierda. Nos aparece una ventana emergente como la que se muestra en la Figura 23.



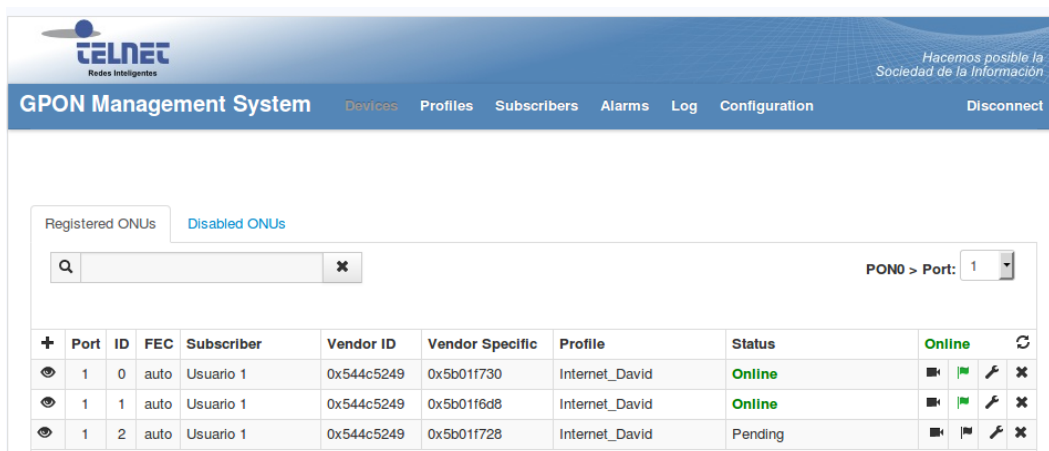
The image shows a screenshot of a web-based configuration window titled "ONU Configuration". The window is divided into several sections for configuring an ONU. On the left side, there are input fields for "ONU ID" (value: 4), "Vendor ID" (value: 0x544c5249), and "Specific ID" (value: 0x5b02e9ca). Below these are two threshold settings: "Signal Degraded Threshold" (Bit Error Rate (BER) - 10, value: -5) and "Signal Fail Threshold" (Bit Error Rate (BER) - 10, value: -3). On the right side, there are dropdown menus for "FEC" (value: default) and "RF PORT (If present)" (value: default). Below these are a "Profile" dropdown (value: Internet\_300Mbps) and a "Subscriber" text field (value: Usuario 1). At the bottom right, there are two buttons: "Close" and "Save".

Figura 23: Ventana para configurar la ONU “ONU Configuration”

En dicha ventana de configuración de la ONU seleccionamos un *Profile*, que se corresponderá con el perfil de servicios que se desee asignar al usuario, en este caso concreto *Internet\_300Mbps*. En el campo *Specific ID* introducimos el parámetro *Vendor Specific*, que hemos comentado anteriormente y finalmente en el campo *Subscriber*, le ponemos un nombre unívoco, en nuestro caso *Usuario 1*, aunque puede ser cualquier otro, *Usuario 2, 3, 4*, etc. El resto de parámetros los dejamos como están en caso de no querer modificar las características de tasa de error de bit (BER, *Bit Error Rate*) o FEC (*Forward Error Correction*). Finalmente guardamos los cambios pinchando en *Save*. Este mismo proceso lo repetimos para todas las ONUs que tenemos conectadas a otro puerto PON y que por lo tanto ha reconocido la red GPON.



- Una vez que tenemos todas las ONUs registradas en el puerto activo, su estado aparece como *Pending*, como podemos ver en la Figura 24.



+	Port	ID	FEC	Subscriber	Vendor ID	Vendor Specific	Profile	Status	Online	
👁	1	0	auto	Usuario 1	0x544c5249	0x5b01f730	Internet_David	Online	■	🔧 ✖
👁	1	1	auto	Usuario 1	0x544c5249	0x5b01f6d8	Internet_David	Online	■	🔧 ✖
👁	1	2	auto	Usuario 1	0x544c5249	0x5b01f728	Internet_David	Pending	■	🔧 ✖

Figura 24: Estado inicial en el que aparecen las ONUs conectadas, estado *Pending*

De este modo, para que su estado pase a *Online*, tenemos que ir a la pantalla de inicio del TGMS y sincronizar de nuevo la OLT con los cambios integrados, en este caso registro de nuevas ONUs. Para ello, se pulsará sobre el botón que nos aparece a la derecha de PON 0, como se observa en la Figura 25, que es el nombre asignado a nuestro OLT.

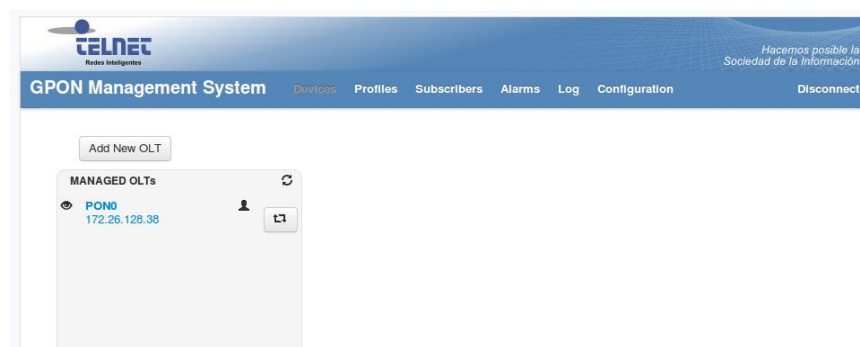


Figura 25: OLT pendiente de sincronización debido al registro de nuevas ONUs

- Volvemos a entrar a analizar el estado de las ONUs una vez realizado el proceso de sincronización, tal y como se ha descrito en el punto 4 y ahora se observa que el estado de todas las ONUs que hemos registrado es *Online*. Es decir, que ahora están registradas y operativas con el perfil de abonado y servicios asignados. Dicha captura de pantalla se muestra en la Figura 26. De este modo ya tenemos

registradas y activadas todas las ONUs que tenemos conectadas a la red GPON al nuevo puerto conectado.

The screenshot shows the 'GPON Management System' interface. At the top, there is a navigation bar with 'GPON Management System' and several menu items: 'Devices', 'Profiles', 'Subscribers', 'Alarms', 'Log', 'Configuration', and 'Disconnect'. Below the navigation bar, there are tabs for 'Registered ONUs' and 'Disabled ONUs'. A search bar is present with a magnifying glass icon and a clear button. To the right, there is a dropdown menu labeled 'PON0 > Port:' with '1' selected. The main content is a table with the following data:

+	Port	ID	FEC	Subscriber	Vendor ID	Vendor Specific	Profile	Status	Online	⌂
👁	1	0	auto	Usuario 1	0x544c5249	0x5b01f730	Internet_David	Online	■	✕
👁	1	1	auto	Usuario 1	0x544c5249	0x5b01f6d8	Internet_David	Online	■	✕
👁	1	2	auto	Usuario 1	0x544c5249	0x5b01f728	Internet_David	Online	■	✕
👁	1	3	auto	Usuario 1	0x544c5249	0x5b02cc68	Internet_David	Online	■	✕
👁	1	4	auto	Usuario 1	0x544c5249	0x5b02e9ca	Internet_David	Online	■	✕

Figura 26: Captura de pantalla donde se observan todas las ONUs que están *Online*

### 3.4.2 Prueba de funcionamiento y análisis de resultados

Para comprobar el correcto funcionamiento del puerto (Port 1), se puede utilizar cualquiera de las ONTs que están conectadas a la red del laboratorio y registradas en TGMS. En este caso la prueba se ha hecho utilizando una ONT de nivel 3 y dicho proceso se describe a continuación.

Para ello, se utilizarán los dos ordenadores Lenovo disponibles en el laboratorio, uno va a ser utilizado como servidor (conectado al ordenador central) y el otro, conectado en el otro extremo de la red (a las ONTs), va a simular a un cliente. La topología resultante es la que se puede ver en la Figura 27.

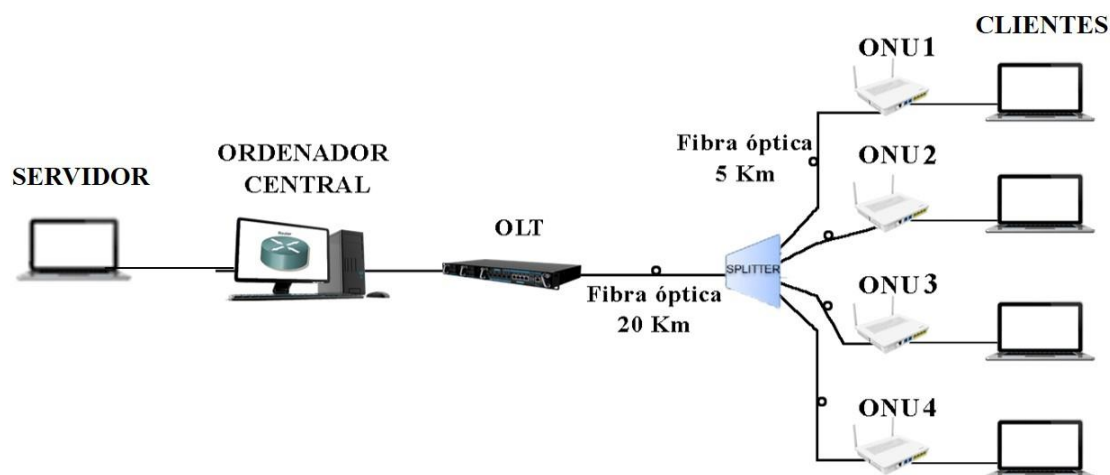


Figura 27: Topología prueba del Puerto 1 del OLT

En el Lenovo utilizado como servidor se abre una terminal, nos identificamos como superusuario y cargamos el archivo *activarServidor.sh* mediante la instrucción `sh activarServidor.sh`. De este modo, se levanta la VLAN 833, se le asigna la dirección IP 10.0.103.60 y se añade la ruta estática necesaria. A continuación, en el ordenador central, de la misma forma se carga el archivo *activar-enrutamiento.sh*, que sirve para levantar la interfaz `enp4s1`, definir la VLAN 833 y hacer que el ordenador central funcione como un router. Posteriormente, hay que entrar al entorno web de TGMS y comprobar que la ONT que se va a utilizar para la prueba tiene asignado un perfil de Internet con un ancho de banda asignado distinto de cero. Unas líneas más abajo, se especificará el ancho de banda asignado para la realización de esta prueba en concreto. A continuación, se conecta el Lenovo que se utiliza como cliente en la línea del laboratorio en la que está conectada la ONT elegida anteriormente, para que de este modo el portátil Lenovo esté conectado a internet a través de dicha ONT. A continuación, se debe configurar el router de dicha ONT de nivel 3 y sincronizar dicha configuración con el OLT a través del TGMS. Para ello, se abre un navegador en el portátil y se va a la dirección 192.168.2.1. El número marcado en negrita varía para cada ONT y es el número con el que se identifica el router de dicha ONT. A continuación, nos encontramos con la interfaz web del router, y en el menú principal hay que entrar en *Device Info* → *WAN* para comprobar que hay un servicio de Internet configurado en el router. Como se puede ver en la Figura 28, donde tenemos la información de los servicios configurados para el router, tenemos que tener la propiedad *Status* en *Connected* para el interfaz asociado a los servicios configurados.

WAN Info													
Interface	Description	Type	VlanMuxId	IPv6	Igmp Pxy	Igmp Enbl	MLD Pxy	MLD Src Enbl	NAT	Firewall	Status	IPv4 Address	IPv6 Address
veip0.1	ipoe_veip0.833	IPoE	833	Disabled	Disabled	Disabled	Disabled	Disabled	Enabled	Disabled	Connected	192.168.0.104	
veip0.2	ipoe_veip0.806	IPoE	806	Disabled	Disabled	Disabled	Disabled	Disabled	Enabled	Disabled	Unconfigured	0.0.0.0	

Figura 28: Captura de pantalla del interfaz de configuración del router de la ONT (*WAN Info*)

Dado que el router de la ONT separa la red en subredes, con el escenario actual, la conexión es viable en el sentido de cliente (lenovo conectado a la ONT) hacia servidor (lenovo conectado al ordenador central hacia fuera de internet). Para poder establecer conexión en el otro sentido y que el cliente sea visible más allá del router, en el ordenador central tenemos que añadir la ruta estática necesaria para que el ordenador central sepa por donde encaminar los paquetes que van con dirección al cliente. Para ello, se utiliza el comando *ip route add* como se indica a continuación *ip route add 192.168.2.0/24 via 192.168.0.104* donde se indica la red que se quiere alcanzar, que en este caso es la que se encuentra al otro lado de la ONT y a través de que dirección la vamos a alcanzar, que en este caso sería la dirección IP del router. Dicha dirección se encuentra en la WAN Info como se observa en la Figura 28.

Para comprobar la conectividad extremo a extremo se utiliza la herramienta *iperf* en cualquiera de las dos direcciones. Los resultados son variables en función del ancho de banda garantizado que esté definido para el perfil de Internet que se asigna a la ONT. Así pues, en caso de que la tasa que se transmite con *iperf* sea inferior a dicho ancho de banda, la capa óptica de la red GPON permite que todo lo que estamos transmitiendo llegue hasta el otro extremo de la red. Sin embargo, si transmitimos una tasa superior al ancho de banda asignado a la ONT, esta misma capa, hace que la tasa de transmisión que llega hasta el otro extremo de la red sea la máxima que tiene definida en su configuración.

Para poder evaluar si el comportamiento del puerto PON 1 es el mismo que el del puerto PON 0 se procede a comparar si la máxima capacidad de la red en ambos casos es semejante para el mismo perfil de abonado y servicios asociados usando la misma ONT. Para ello, en TGMS, tenemos que asociar a todas las ONT menos a una el perfil de *Internet Vacío* (sin ancho de banda asignado) y en el perfil de la ONT que vamos a utilizar le asignaremos un perfil de Internet con un ancho de banda igual al máximo que permite TGMS. El máximo que permite TGMS en ambos canales de transmisión está configurado para que sea de 1.24 Gbps. Este máximo coincide con el marcado por el

estándar GPON para el canal ascendente, para el canal descendente el máximo que dicta el estándar es 2.5 Gbps. De este modo se evitan errores de configuración que impidan la puesta en marcha de la red. Los resultados que se obtienen para la misma ONT con el mismo servicio en ambos puertos (PON 0, PON 1) se pueden ver en Tabla 3 y son acordes con los observados en la Tabla 1, en la que tenemos las máximas tasas que se obtienen entre las tarjetas de red del ordenador central.

Puerto/Sentido de transmisión	Upstream	Downstream
PON 0	761 Mbps	630 Mbps
PON 1	779 Mbps	672 Mbps

Tabla 3. Resultados del ancho de banda ofrecido a la misma ONT cuando se conecta a ambos puertos para el mismo servicio y perfil de abonado asociado.

A tenor de los resultados observados en la Tabla 3, que son muy semejantes, se puede concluir que el funcionamiento del puerto PON 1 del OLT es el mismo que el del puerto PON 0.

### 3.5 Registro y activación de ONTs de nivel 2

En este apartado de la memoria se describe el proceso de registro y activación de dos nuevas ONTs de nivel 2 preparadas para su instalación e integración en la red GPON del laboratorio [16]. La principal diferencia entre una ONT de nivel 2 y una de nivel 3 se encuentra en que las ONTs de nivel 2 actúan como un switch, sin tener funcionalidades de router, por lo que son transparentes para el resto de dispositivos de red. Por este motivo, es necesario integrar algún router bien físico o virtual que realice este tipo de funcionalidades. En nuestro caso en concreto, hemos hecho uso de la tecnología OVS (*OpenVirtual Switch*) es decir un switch virtual que se encargue de enrutar los paquetes hacia donde corresponda. Cabe destacar que OVS es compatible con SDN y con el estándar OpenFlow, que es el usado en el escenario SDN desplegado en nuestra red GPON. Para dar mayor flexibilidad y portabilidad entre máquinas a los OVS instalados, se ha optado por desplegar los OVS asociados a las ONTs en máquinas virtuales mediante la herramienta VirtualBox [17], de modo que si, queremos desplegar OVS para conectarlo con una ONT de nivel 2, tan solo haríamos una copia de la máquina virtual y la configuraríamos de forma adecuada. Sin embargo, el OVS que se encuentra en el lado del ordenador central y que se conecta al OLT para gestionar el tráfico que viene de

Internet hacia las ONTs en el sentido *downstream* se optó por instalarlo en el ordenador y no en máquinas virtuales, tal y como se ha explicado en apartados anteriores.

### **3.5.1 Proceso de activación y análisis de rendimiento de ONTs de nivel 2**

Después de conectar cada ONT de nivel 2 a la red GPON, tenemos que registrar dicha ONT siguiendo los pasos que se han descrito anteriormente en este documento. Como la solución SDN-GPON ya tenía integrada una ONT de nivel 2 y el objetivo en primer término es probar que las ONTs nuevas que se han registrado funcionan correctamente, se va a utilizar el OVS que ya está instalado en el escenario SDN-GPON inicial a las nuevas ONTs de nivel 2.

Para ello, se sigue el siguiente proceso. En primer lugar, en el ordenador central se tiene que activar el OVS que allí se encuentra instalado, el que gestiona el tráfico que viene de Internet hacia dentro de la red GPON (*dowstream*). Para ello, abrimos una terminal y nos identificamos como superusuarios, cargamos el archivo *activar-enrutamineto.sh* y activamos el OVS tecleando *sh ovs.sh*. Ahora, generamos un flow por defecto para que el sistema no se caiga, tecleando *sh ovs\_2.sh*. Para comprobar que el flow está activo tecleamos *ovs-ofctl -O OpenFlow13 dump-flows br0*, y en caso de no estar activo, volvemos a teclear *sh ovs\_2.sh*.

A continuación, en el ordenador de la torre se activa el OVS que se conecta con la nueva ONT de nivel 2 y que está instalado dentro de una máquina virtual. Para activarlo, tenemos que levantar la VLAN 833 (VLAN asignada a la red GPON para los servicios), cargando el archivo *configuraciónVLAN.sh*, cuyo código completo se muestra en la Figura 29.

```
ifconfig enp3s6 0
vconfig add enp3s6 833
ifconfig enp3s6.833 up
dhclient enp3s6.833

#vconfig add enp3s6 806
#ifconfig enp3s6.806 up
#dhclient enp3s6.806
```

Figura 29: Código del archivo configuracionVLAN.sh

A continuación, arrancamos la máquina virtual, nos identificamos como superusuarios y tecleamos *sh script.sh*, cuyo código podemos ver en la Figura 30, de este

modo activamos el OVS. Ahora, también generamos un flow por defecto tecleando *sh script2.sh*, cuyo código se muestra en la Figura 31 y volvemos a comprobar que está activo de la misma forma que en el ordenador central.

```
/usr/local/share/openvswitch/scripts/ovs-ctl start
ifconfig enp0s3 0
ifconfig enp0s8 0
ifconfig br0 up
ifconfig br0 192.168.0.2/24
route add -net default gw 192.168.0.1 dev br0
```

Figura 30: Código del archivo script.sh

```
ovs-ofctl -O OpenFlow13 add-flow br0 priority=20,actions=normal
```

Figura 31: Código del archivo script2.sh

De nuevo en el ordenador central, tenemos que activar el controlador ODL. No hay más que arrancar la máquina virtual de nombre controller, identificarse como superusuario, ir a la ruta */home/controller/Descargas/karaf-0.7* y teclear *bin/karaf*. Pasados unos minutos, tanto en el ordenador central como en el ordenador de la torre comprobamos que se han generado los flows que hay configurados en el controlador.

Finalmente, conectamos uno de los Lenovo al ordenador de la torre y el otro, que va a actuar como servidor al ordenador central, cargando el archivo *activarServidor.sh* como en otras ocasiones. Podemos ver la topología resultante en la Figura 32.

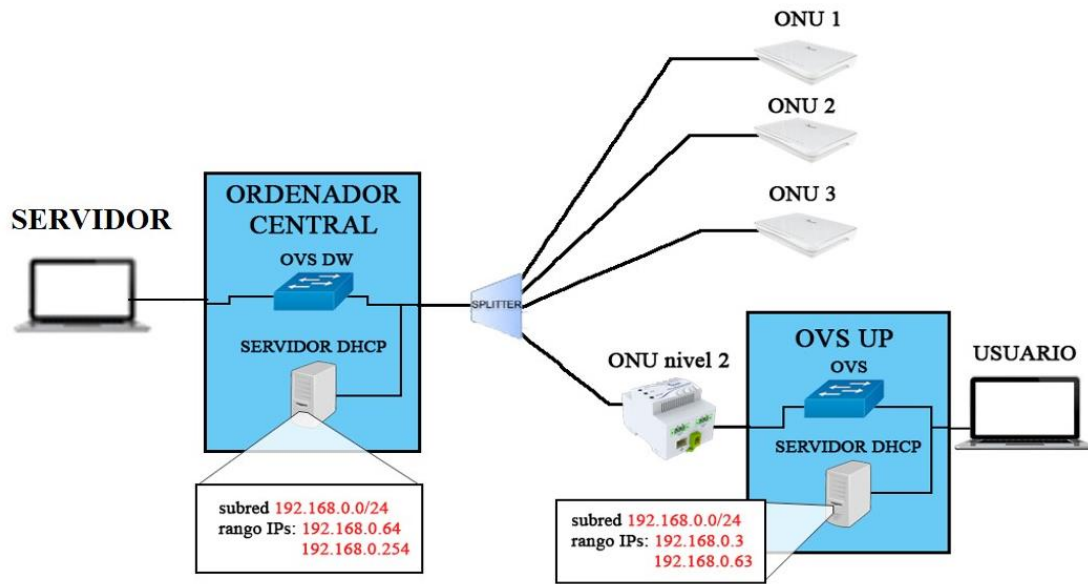


Figura 32. Topología para la prueba de las ONTs de nivel 2.

En la Tabla 4 se muestran los resultados de la máxima capacidad que soporta la red tanto para las dos ONTs nuevas de nivel 2 como para la que ya estaba operativa en el laboratorio, y así comparar su comportamiento y rendimiento. Tal y como se observa en la Tabla 4, las tres ONTs de nivel 2 presentan un comportamiento muy similar, que también es muy semejante al que presentan las ONTs de nivel 3.

ONT/Sentido de transmisión	Upstream	Downstream
ONT id 0x5b02e9ca	782 Mbps	625 Mbps
ONT id 0x5b03fc7f	769 Mbps	603 Mbps
ONT id 0x5b03fc82	781 Mbps	633 Mbps

Tabla 4. Resultados de rendimiento de las ONTs de nivel 2 conectadas a la red SDN-GPON.

Como ya se ha comentado anteriormente para llevar a cabo esta prueba se ha utilizado el mismo OVS para comparar el funcionamiento de las tres ONTs. Sin embargo, sería adecuado replicar un OVS para cada ONT y que el controlador los identifique como diferentes con unos simples ajustes. De este modo, tendríamos activas y operativas varias ONTs de nivel 2, cada una con su OVS y router activado.



### 3.5.2 Proceso de despliegue de OVS's en máquinas virtuales

Como ya se ha comentado con anterioridad, el OVS que controla el flujo ascendente (conectado a la ONT), se encuentra instalado dentro de una máquina virtual en un ordenador de sobremesa. Pero además del OVS, en dicha máquina virtual también tenemos instalado un servidor DHCP y la aplicación para la gestión de servicios y de la red residencial del usuario, tal y como se puede ver en la Figura 33. Por ello, resulta muy importante clonar dicha máquina virtual sin perder nada de lo que tenemos instalado dentro y replicar de un modo sencillo el OVS y las herramientas software necesarias.

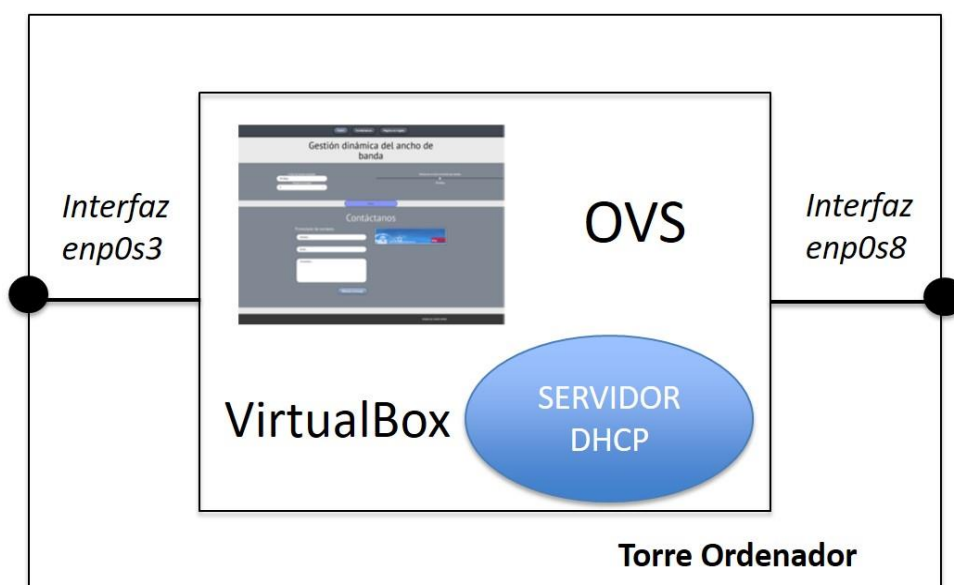


Figura 33: Esquema de la máquina virtual del ordenador de la Torre

Para hacerlo, en primer lugar, accedemos a Virtual Box y pinchamos en *Archivo* → *Exportar servicio virtualizado*. Seleccionamos *Modo experto* y aparecerá una ventana semejante a la que se muestra en la Figura 34. No hay más que elegir la ubicación del archivo que se va a generar y pinchar en *Exportar* dejando el resto de opciones por defecto.

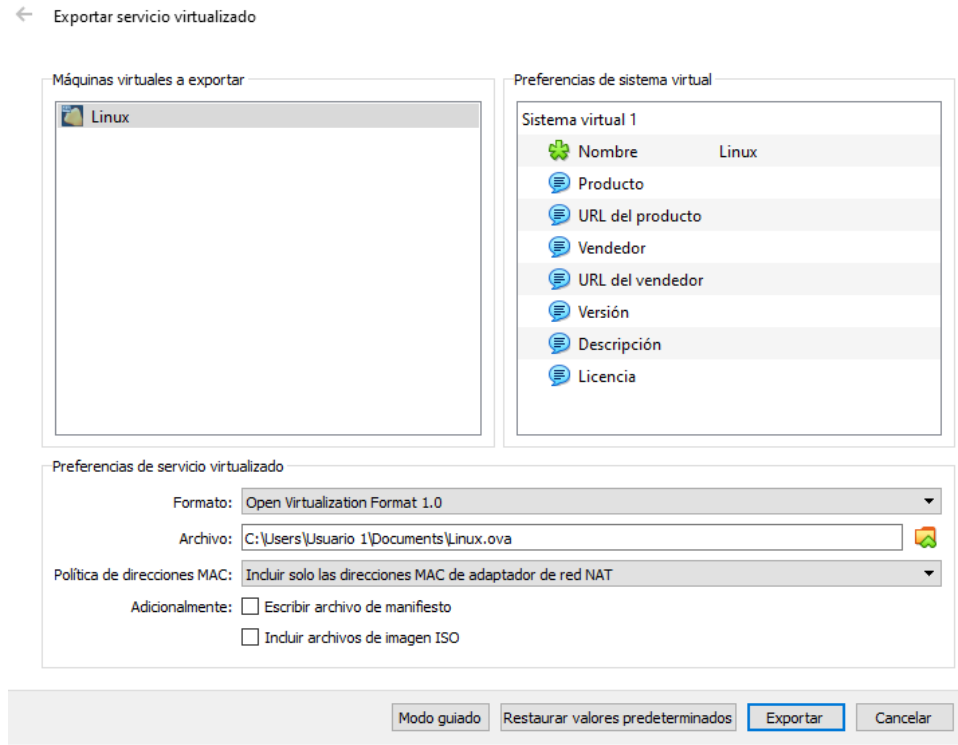


Figura 34: Proceso para exportar un servicio virtualizado de una máquina virtual

Pasados unos minutos, cuando se haya generado el archivo con extensión *.ova* hay que importarlo en el equipo que se desee. Para ello, vamos a *Archivo* → *Importar servicio virtualizado*, seleccionamos el archivo que hemos importado anteriormente y pinchamos en *Importar*.

Además de clonar la máquina virtual, y puesto que se instalará en un equipo diferente, hay que copiar también el archivo *configuracionVLAN.sh* y modificarlo, adecuándolo a las interfaces del nuevo equipo. Posteriormente, en Virtual Box tenemos que seleccionar el adaptador de red que corresponda con las interfaces del nuevo equipo. En la Figura 35, la Figura 36 y la Figura 37 se puede ver cual sería el resultado final correspondiente para el nuevo ordenador donde se ha instalado la copia de la máquina virtual tanto para el archivo *configuracionVLAN.sh* como para la nueva máquina virtual.

```
configuracionVLAN.sh (admin:///root)
Archivo  Editar  Ver  Buscar  Herramientas  Documentos  Ayuda
configuracionVLAN.sh x
ifconfig enp7s0 0
vconfig add enp7s0 833
ifconfig enp7s0.833 up
dhclient enp7s0.833

#vconfig add enp7s0 806
#ifconfig enp7s0.806 up
#dhclient enp7s0.806
```

Figura 35: Archivo *configuracionVLAN.sh* para el ordenador nuevo

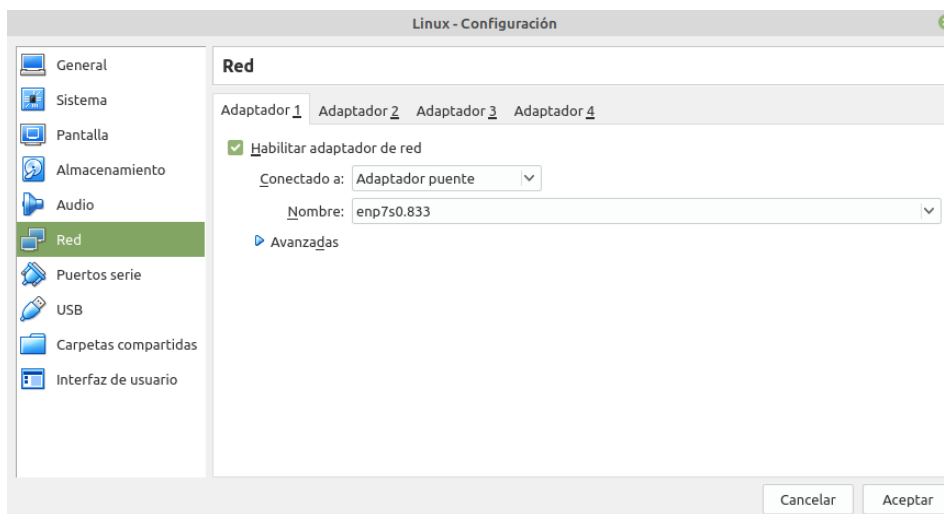


Figura 36: Adaptador de red 1 para la máquina virtual del nuevo OVS

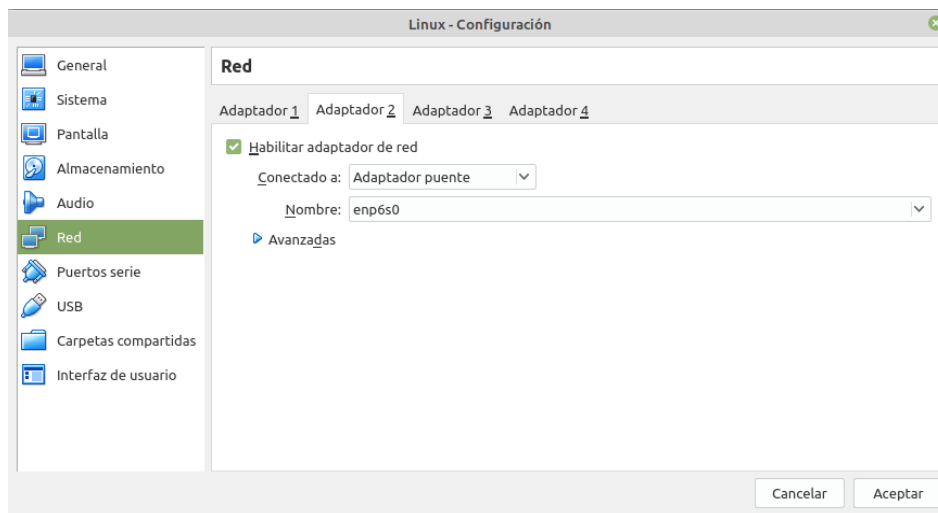
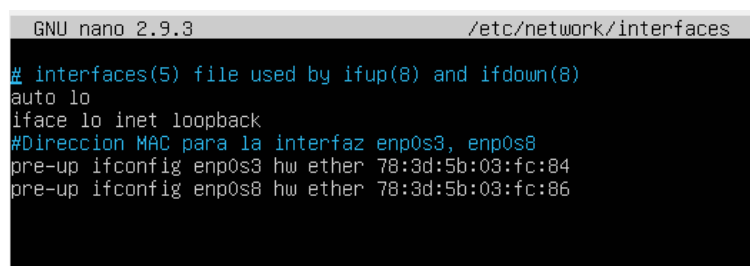


Figura 37: Adaptador de red 2 para la máquina virtual del nuevo OVS

La copia que se ha hecho es un clon exacto de la otra máquina virtual, por lo que el controlador ODL lo va a identificar como el mismo OVS, esto es, con el mismo identificador, por lo que se tienen que modificar las direcciones MAC de las interfaces que tenemos configuradas en dicha máquina virtual. Para ello, en la máquina virtual,

tenemos que editar el archivo `/etc/network/interfaces` y añadir la siguiente instrucción: `pre-up ifconfig "nombre de la interfaz" hw ether "dirección MAC"`. Para la interfaz `br0`, no es necesario indicar dirección MAC, ya que automáticamente se asignará la misma dirección que para la interfaz `enp0s3`. Las direcciones MAC elegidas no son aleatorias, sino que coinciden en parte con el identificador que proporciona TGMS para cada ONT. La regla para asignar dichas direcciones es la siguiente, en primer lugar, dado que los identificadores de TGMS contienen guiones para separar los dígitos en grupos de dos, tenemos que sustituir estos guiones por ":". A continuación, como es sabido que las direcciones MAC están formadas por doce dígitos en formato hexadecimal, tenemos que quedarnos únicamente con parte del identificador de TGMS, ya que estos cuentan con 16 dígitos. Para ello, tenemos que sustituir los 8 primeros dígitos, además de las separaciones, del identificador de TGMS por "78:3d". De este modo, ya contamos con doce dígitos separados en grupos de dos por ":". Pero todavía necesitamos hacer un último cambio en la dirección, ya que ahora mismo lo que tenemos es la dirección MAC de la ONT, por lo que, por último, tenemos que sustituir el último dígito de la dirección por otro, en este caso, aleatoriamente, siempre teniendo en cuenta, que las direcciones MAC están en formato hexadecimal. Estas reglas para la asignación de las direcciones MAC de las interfaces del OVS, nos sirven para facilitar la programación de ciertos aspectos del programa en Python que controla el funcionamiento de la red SDN-GPON. En la Figura 38, se puede ver el resultado final del archivo.



```
GNU nano 2.9.3 /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback
#Dirección MAC para la interfaz enp0s3, enp0s8
pre-up ifconfig enp0s3 hw ether 78:3d:5b:03:fc:84
pre-up ifconfig enp0s8 hw ether 78:3d:5b:03:fc:86
```

Figura 38: Archivo interfaces de la máquina virtual del nuevo OVS

Como ya se ha comentado anteriormente, en la máquina virtual también se encuentra instalado un servidor DHCP, por lo que también debemos modificarlo. Para ello, modificamos el archivo que se encuentra en la siguiente ruta `/etc/dhcp/dhcpd.conf`. Lo único que debemos modificar en dicho archivo es el rango de direcciones IP que le vamos a otorgar al servidor DHCP. Como se puede ver en la Figura 39, el rango asignado corresponde con la subred `192.168.0.64/26`, aunque realmente la subred a la que pertenece es la `192.168.0.0/24`; de este modo todos los OVSs pertenecen realmente a la

misma subred, definida en el servidor DHCP instalado en el ordenador central, aunque el rango asignado a cada servidor DHCP de las ONTs es diferente y corresponde a una subred de 26 bits. Por este motivo, en el servidor DHCP del ordenador central también tenemos que modificar el rango asignado, eliminando la parte del rango que le hemos dado al servidor DHCP del nuevo OVS.

```
GNU nano 2.9.3 /etc/dhcp/dhcpd.conf
# subnet 10.17.224.0 netmask 255.255.255.0 {
#   option routers rtr-224.example.org;
# }
# subnet 10.0.29.0 netmask 255.255.255.0 {
#   option routers rtr-29.example.org;
# }
# pool {
#   allow members of "foo";
#   range 10.17.224.10 10.17.224.250;
# }
# pool {
#   deny members of "foo";
#   range 10.0.29.10 10.0.29.230;
# }
#}
subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.67 192.168.0.126;
  option domain-name-servers 8.8.8.8,8.8.4.4;
  option routers 192.168.0.1;
  option broadcast-address 192.168.0.255;
  default-lease-time 600;
  max-lease-time 7200;
}
```

Figura 39: Servidor DHCP del nuevo OVS

La última modificación que debemos realizar con respecto a las direcciones, debemos llevarla a cabo en el archivo *script.sh*. En dicho archivo, tenemos que modificar la dirección IP asignada a la interfaz *br0*. La dirección asignada tiene que estar fuera del rango del servidor DHCP y como en otros casos sigue una regla que facilite posteriormente la programación del código en Python. Esta regla es simple, no tenemos más que coger la dirección de la subred, en este caso *192.168.0.64* y sumarle dos, de modo que nos queda la dirección *192.168.0.66*. Esta dirección es la que tenemos que teclear cuando posteriormente, queramos acceder a la página web que también se encuentra en la máquina virtual. En la Figura 40, vemos el resultado final del archivo.

```
|/usr/local/share/openvswitch/scripts/ovs-ctl start
ifconfig enp0s3 0
ifconfig enp0s8 0
ifconfig br0 up
ifconfig br0 192.168.0.66/24
route add -net default gw 192.168.0.1 dev br0
```

Figura 40: Archivo *script.sh* del nuevo OVS

Por último, también hay que realizar algunos cambios en el código PHP de la aplicación web del usuario. Para acceder a los archivos que contienen el código PHP, tenemos que ir a la ruta *var/www/html/php*. En el archivo *abrirSocket.php*, tenemos que sustituir la dirección del socket del servidor por la que le hayamos asignado a la interfaz *br0*, que en este caso es *192.168.0.66*. En los archivos *gestionarRedResidencial.php* y *servicio.php*, tanto en su versión en español, como en su versión en inglés, tenemos que sustituir la ONT a la que nos referimos por la ONT que corresponde con el OVS. En este caso sería la *54-4c-52-49-5b-03-fc-82*. Después de todos estos cambios ya tenemos nuestro nuevo OVS operativo, siendo totalmente independiente del que teníamos anteriormente.

### **3.6 Conclusiones**

En este capítulo de la memoria, después de describir el escenario SDN inicial que estaba implementado en la red GPON del laboratorio, se han expuesto los resultados obtenidos al llevar a cabo las diferentes pruebas de los equipos integrados en la red. Se ha podido concluir que, con las conexiones actuales implementadas el comportamiento no era el más óptimo posible. Para mejorar dicho comportamiento, se han tomado las decisiones de cambio oportunas para mejorar el rendimiento de la red GPON, presentando, por último, la topología final con dichos cambios realizados. También se ha podido comprobar que el funcionamiento del puerto 1 del OLT, es el mismo que el del puerto 0. Respecto a la integración de nuevas ONTs en la red, dos ONTs de nivel 2 en este caso, se han descrito los pasos a seguir para conseguir su correcto registro y se han establecido las pautas necesarias para instalar los OVSs asociados a dichas ONTs de nivel 2 para su gestión mediante tecnologías SDN. Este paso es necesario en caso de querer un funcionamiento simultáneo de varias ONTs de nivel 2, acercando en mayor medida la red del laboratorio a una red real controlable mediante SDN.

# 4

## **Automatización de la red SDN-GPON**

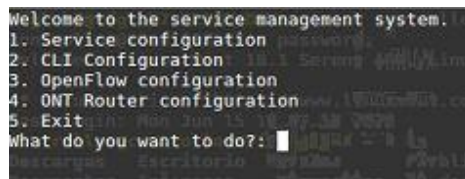
### **4.1 Introducción**

A lo largo de este capítulo de la memoria se va a describir el proceso de automatización que se ha llevado a cabo sobre la red SDN-GPON con el objetivo de que sea fácilmente escalable sin necesidad de grandes cambios. En primer lugar, se va a hacer un análisis del flujo del programa que controla la red, haciendo especial mención sobre aquellas funcionalidades más importantes ya implementadas, como son la creación de servicios temporales, la gestión de la red residencial y el algoritmo de configuración dinámica de servicios. En concreto, nos centraremos en intentar automatizar el código lo máximo posible con la finalidad de integrar nuevos dispositivos y ONTs de un modo transparente al usuario. Posteriormente, se van a describir las funciones desarrolladas para conseguir la automatización buscada y se van a mostrar los resultados de una prueba que demuestra que se ha alcanzado el objetivo pretendido. Por último, se va a realizar un análisis más detallado del comportamiento del algoritmo de configuración dinámica de servicios.

### **4.2 Descripción del flujo del programa que controla la red SDN-GPON**

Para poder acometer la actualización y automatización del programa que controla la red SDN-GPON, es importante conocer muy bien el flujo de dicho programa, por lo que a continuación se van a describir brevemente todas las opciones del programa global poniendo especial atención en aquellos puntos en los que posteriormente se actuará de cara a su automatización.

De este modo, cuando se inicia el programa, nos encontramos con un menú inicial con cinco opciones, como se puede ver en la Figura 41 y posteriormente se describirán brevemente:



```
Welcome to the service management system.
1. Service configuration
2. CLI Configuration
3. OpenFlow configuration
4. ONT Router configuration
5. Exit
What do you want to do?:
```

Figura 41: Menú inicial con las funcionalidades implementadas

1. *Service configuration*: esta opción permite por un lado visualizar la lista de los servicios configurados en la red GPON. Pero además permite crear nuevos servicios, introduciendo el tipo de servicio que vamos a crear (Internet o Video), la VLAN sobre la que se va a definir dicho servicio y finalmente el ancho de banda garantizado y en exceso para ambos sentidos de transmisión. También, permite modificar algún parámetro de los servicios ya creados y por último borrar alguno de los servicios existentes en la red.
2. *CLI Configuration*: opción que permite cambiar la configuración de la red a través de CLI. Consta de dos opciones, una para cargar un servicio de los ya existentes sobre alguna de las ONTs que se encuentran conectadas a la red y otra para eliminar el servicio que tenga asignado cualquiera de las ONTs conectadas.
3. *OpenFlow configuration*: esta opción nos lleva a un submenú, que podemos ver en la Figura 42, en el que se muestran una serie de funcionalidades. Entre concreto, cargar o eliminar servicios OpenFlow sobre alguna de las ONTs, mostrar los OVS activos en la red en ese momento, iniciar la ejecución del algoritmo de configuración dinámica de servicios, iniciar la ejecución de la aplicación para la gestión de servicios temporales y finalmente iniciar la ejecución de la aplicación que controla la red residencial. Estas tres últimas opciones se describirán con más detalle en puntos posteriores. Para simplificar el menú y adaptar la aplicación a una gestión más real, posteriormente se van a juntar las opciones 5 y 6 en una sola, ya que ambas están muy ligadas, pues el usuario tiene acceso a ellas a través de una misma aplicación web.



```
You have chosen: OpenFlowConfig
1. Attach OpenFlow service to ONT
2. Detach OpenFlow service from ONT
3. Show active OVS
4. Automatic Service Configuration Policies
5. Start application web customize
6. Start Residential network customizer
7. Go back
What do you want to do?: 
```

Figura 42: Menú Openflow Configuration

4. *ONT Router configuration*: opción que nos permite crear, borrar, modificar servicios de internet o video sobre una determinada ONT y crear o borrar políticas de enrutamiento. Todo ello utilizando archivos de configuración XML.
5. *Exit*: opción que permite finalizar la ejecución del programa.

A continuación, en los siguientes apartados se va a realizar una descripción más profunda de algunas de las opciones relacionadas con el control y configuración de la red GPON mediante OpenFlow.

#### **4.2.1 Descripción y diagrama de flujo del algoritmo de configuración dinámica de servicios en la red SDN-GPON**

El algoritmo se encarga de actualizar el ancho de banda entregado a los clientes en tiempo real y de forma transparente al usuario. El ancho de banda se actualiza en función de la demanda en tiempo real de los usuarios, hasta un máximo establecido que varía según la tarifa contratada, aplicando por lo tanto un modelo de tarifas flexibles en el que un usuario contrata un servicio básico más un extra máximo incluido que se le dará cuando éste lo pida y haya recursos disponibles suficientes en la red. La evolución en tiempo real del ancho de banda demandado, se hace a través del control de las estadísticas de OpenFlow.

Como se puede ver en la Figura 43, para activar el algoritmo de configuración dinámica de servicios, tenemos que seleccionar en primer lugar la Opción 3 del menú inicial (*OpenFlow configuration*) y posteriormente la Opción 4 del submenú. Simultáneamente, a través de una conexión *ssh* con el controlador ODL, ejecutamos el archivo *socky.py*. En este momento, el programa comienza su ejecución, ejecutando una serie de métodos que se encuentran en el archivo *statsOVS.py*. En primer lugar, se abre un socket, escuchando en las interfaces *br0* y *enp0s31f6.833*. La interfaz *br0*, es el puente

creado por el OVS del ordenador central y nos va a permitir descubrir la dirección, la interfaz y el puerto utilizados por cada uno de los OVS situados al lado de los clientes. Por otro lado, la interfaz *enp0s3If6.833*, que va dirigida hacia el exterior de la red, donde se encuentra el controlador ODL, que nos permite descubrir la dirección, interfaz y puerto utilizados por el OVS del ordenador central. A continuación, se realiza una llamada al método *StatsOVS()*, que a su vez llama a los métodos *staticsThread()* y *Controller OVS()*. Dichos métodos, se ejecutan dentro de un bucle infinito, por lo que su ejecución no se detendrá a no ser que se produzca un error.

Por un lado, el método *staticsTread()* es el encargado de ir llenando la ventana con los datos de la evolución en tiempo real del ancho de banda demandado por la ONT y poniendo los mensajes en la cola, que contienen los datos necesarios para que posteriormente se pueda actualizar el servicio. Para ello, se abre un socket asociado a la dirección IP del ordenador central (10.0.103.48) y se recogen los paquetes que llegan. Estos paquetes contienen únicamente las estadísticas de los *meters*, ya que el mensaje OpenFlow completo es filtrado en el controlador haciendo uso del script *socky.py*, que se encuentra en la máquina virtual donde está instalado el controlador, enviando al ordenador central únicamente las estadísticas OpenFlow. Mediante un bucle se recorren los *meters* que contienen, calculando los bytes totales, finales y el tiempo de cada paquete. Tras esto, se añaden los valores a la ventana, que se crea si inicialmente no existe. A partir de que la ventana se llene por primera vez, ya se comienza a calcular el ancho de banda total, medio y final y en caso de que el ancho de banda medio sea superior al actual, se añade a la cola el mensaje con los datos necesarios para que se pueda actualizar el servicio al nuevo ancho de banda demandado por el usuario residencial. Se trata de una ventana deslizante, de modo que cuando la ventana está llena, se van añadiendo muestras nuevas con estadísticas openflow más actuales y se van eliminando las más antiguas.

Por otro lado, el método *Controller OVS()*, se encarga de ir recogiendo los mensajes de la cola para ir actualizando el ancho de banda. Para ello, el algoritmo define los porcentajes de ancho de banda en exceso que se pueden dar al usuario en función del servicio básico contratado, en concreto, un tanto por ciento de dicho ancho de banda. A continuación, se recogen los mensajes de la cola, obteniendo sus parámetros y determinando la ONT activa para llamar al método *GetOVS()*, que devuelve la dirección

MAC de la ONT y el identificador y la MAC de la interfaz del OVS correspondiente. Posteriormente, se conecta con la base de datos para obtener el ancho de banda del servicio básico y comprobar si existe un servicio temporal activo y calcula el ancho de banda máximo que corresponde a la tarifa flexible contratada. Finalmente, en caso de que el ancho de banda demandado en tiempo real por el usuario (ONT) sea superior al del servicio activo, se comprueba si la red dispone de ancho de banda suficiente a través del método *BandComp()*, que suma el ancho de banda de todos los servicios activos, comparándolo con el máximo ancho de banda disponible en la red tanto para el canal de subida como el de bajada y se crea el servicio dinámico correspondiente, eliminando previamente los servicios dinámicos que hubiese activos. En caso de que el ancho de banda demandado sea superior al máximo establecido para la tarifa contratada, se concederá dicho ancho de banda máximo. Por el contrario, si el ancho de banda demandado es inferior al servicio básico activo, se eliminan los servicios dinámicos. Por último, en ambos casos, se actualiza la base de datos.

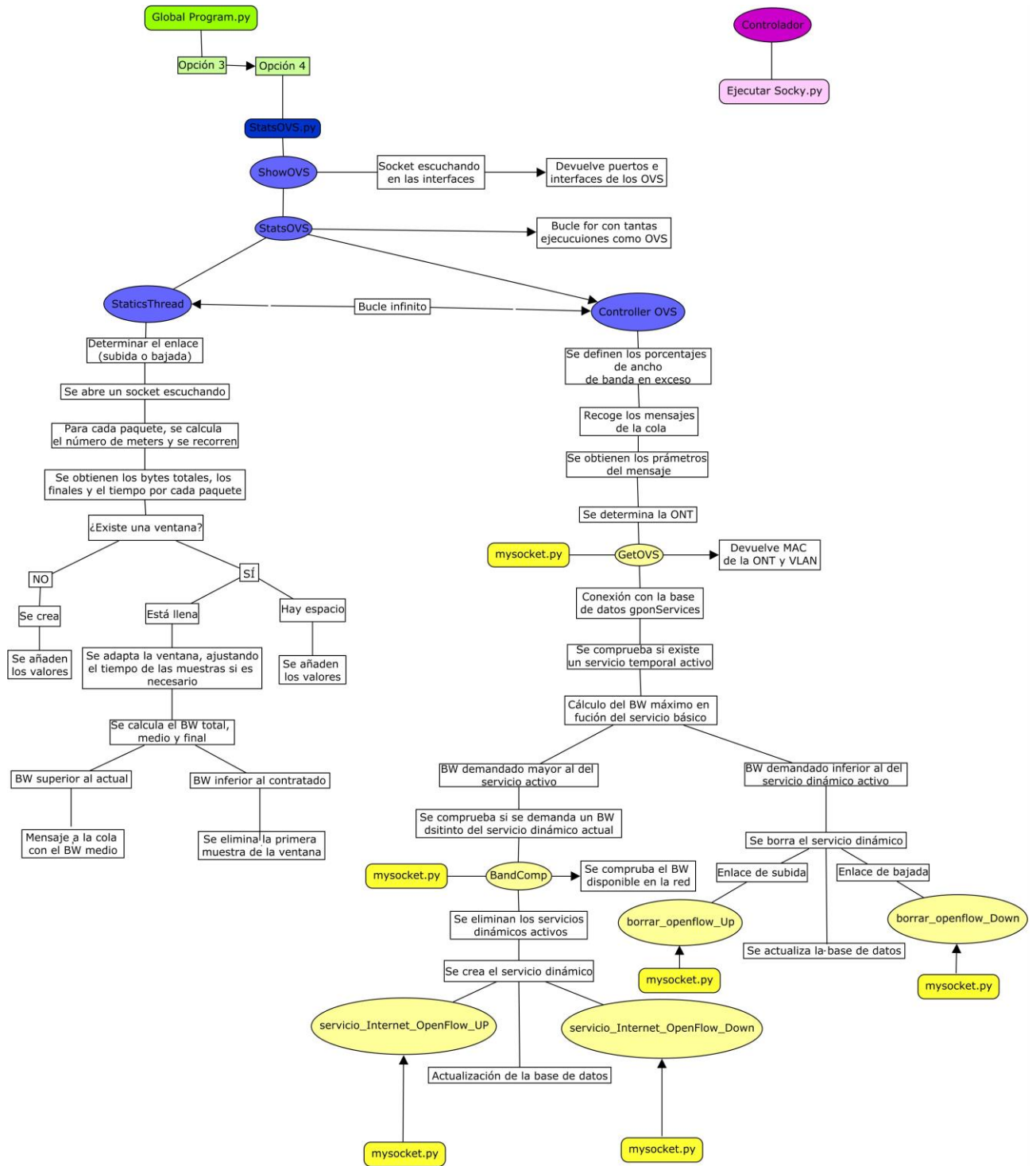


Figura 43: Diagrama de flujo del algoritmo de configuración dinámica de servicios

### **4.2.2 Descripción y diagrama de flujo de la aplicación para la gestión de servicios temporales**

Como se puede ver en la Figura 44, para ejecutar la aplicación para la gestión de servicios temporales, tenemos que seleccionar la Opción 3 del menú inicial (*OpenFlow configuration*) y posteriormente la Opción 5 del submenú. En primer lugar, accedemos al método *sockMain()*, que se encuentra en el archivo *StatsOVS.py*, desde donde se llama al método *mysocket()* que se encuentra en el archivo *mysocket.py*, que es donde se encuentra el desarrollo real del programa.

En primer lugar, se abre un socket escuchando en el puerto indicando y a través de un bucle infinito se van recogiendo las peticiones de los usuarios. Cada vez que llega una petición, se leen y decodifican los datos recibidos, se separan las partes del mensaje y se obtiene la dirección MAC del OVS de upstream, asociándolo con la correspondiente ONT. Tras esto, se llama al método *BandComp()*, que se encarga de comprobar si existe ancho de banda disponible en la red, sumando el ancho de banda de todos los servicios activos, comprobando finalmente que esta suma no exceda el máximo de la red tanto para el canal de bajada como el de subida. Posteriormente, y en el caso de que haya suficientes recursos de ancho de banda en la red, se crea el servicio temporal correspondiente, haciendo uso del parámetro *hard\_time out*, que indica que tras el tiempo indicado se elimina el flow automáticamente, esto es, el servicio temporal contratado por el usuario. Por último, se actualiza este nuevo servicio temporal en la base de datos. Sin embargo, en caso de que la red no tenga recursos suficientes, simplemente se responde negativamente al usuario con un mensaje enviado a través del socket.

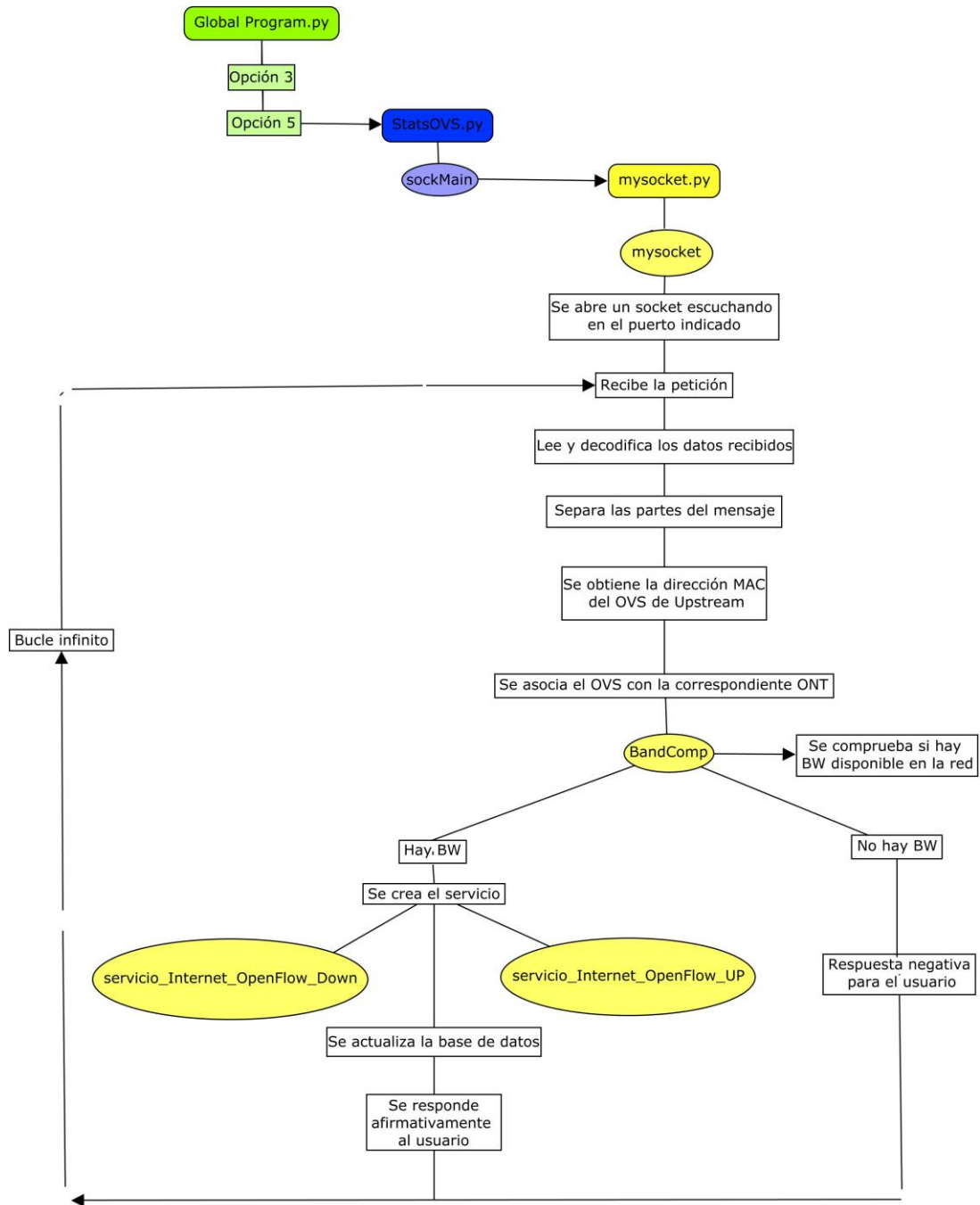


Figura 44: Diagrama de flujo de la aplicación para la gestión de servicios temporales

### 4.2.3 Descripción y diagrama de flujo de la aplicación para la gestión de la red residencial

Como se puede observar en la Figura 45, para activar la aplicación para la gestión de la red residencial, hay que seleccionar la Opción 3 del menú inicial (*OpenFlow configuration*) y posteriormente la Opción 6 del submenú. Una vez hecho esto, la aplicación se pone en marcha. En primer lugar, en el método *residMain()* del archivo

*StatsOVS.py* se llama al método *redResidencial()* que se encuentra en el archivo *redResidencial.py*, donde se abre un socket que escucha en el puerto indicado. A continuación, dentro de un bucle infinito, se van recibiendo las peticiones de los usuarios, decodificando cada mensaje de forma que se obtienen los datos necesarios, incluidos los identificadores de la ONT y el dispositivo sobre el que se quiere imponer una restricción en sus recursos. Las órdenes que se pueden recibir en dichos mensajes son tres, GET, BW y PRI y en función de dichas órdenes se actúa de una forma u otra.

La función de la orden GET, es devolver los servicios activos de la red residencial para el dispositivo seleccionado. Esta orden, se envía desde la aplicación web, de forma transparente al usuario, cuando el cliente desea conocer las restricciones de prioridad o ancho de banda asociados a un determinado dispositivo. Para ello, en primer lugar, se conecta con la base de datos, obteniendo el id del servicio básico, el ancho de banda en ambos sentidos y los servicios de la red residencial activos. Con esta información se prepara el mensaje de respuesta que se envía hacia la aplicación web a través del mismo medio por el que se reciben las peticiones.

Para el caso de las órdenes BW y PRI, los primeros pasos son comunes, de modo que en primer lugar se conecta con la base de datos, para obtener el id del servicio básico y el ancho de banda máximo en ambos sentidos. La orden BW, permite al usuario establecer una restricción de ancho de banda máximo para un determinado dispositivo desde la aplicación web que tiene el usuario para gestionar la red residencial, y para ello las operaciones a realizar se diferencian en función del ancho de banda máximo seleccionado para el dispositivo en concreto. Si el ancho de banda máximo seleccionado es igual al servicio básico contratado, quiere decir que, en realidad, no se va a aplicar ninguna restricción sobre el dispositivo seleccionado, por lo que no es necesario que se cree un nuevo servicio en caso de que no exista, de modo que el ancho de banda disponible se repartirá entre todos los dispositivos de la red según su ancho de banda demandado (en este caso todos los dispositivos de la red residencial son iguales). Para aceptar esta nueva restricción, en caso de que exista un servicio de la red residencial para ese dispositivo en concreto, este debe ser eliminado, y para ello en primer lugar se obtienen los parámetros necesarios de la ONT a través del método *GetOVS()* y con estos datos y con los que ya se tienen del dispositivo que realiza la petición, se eliminan los servicios de la red residencial (mediante orden *openflow*). En ambos casos, se le da,

finalmente, una respuesta afirmativa para el usuario. Por otro lado, si el ancho de banda seleccionado es superior al ancho de banda del servicio básico contratado, la respuesta es negativa, ya que no se puede establecer una restricción de ancho de banda por encima del que tiene contratado. Por último, si la restricción de ancho de banda solicitada es inferior al ancho de banda del servicio básico, se debe crear el servicio con el ancho de banda seleccionado, de modo que se tienen que eliminar los servicios de la red residencial que hubiese creados con anterioridad para el dispositivo. Para ello, como en otros casos, se llama al método *GetOVS()* y se comprueba si existe un servicio de la red residencial creado (servicio openflow), para eliminarlo si ya existe y posteriormente crear el servicio correspondiente y dar la respuesta afirmativa al usuario.

Por último, la orden PRI, permite establecer tres órdenes de prioridad para los dispositivos conectados a la red residencial del abonado, HIGH, MED y LOW. Como en otros casos, lo primero que se hace es obtener los datos necesarios de la ONT correspondiente y comprobar si existe algún servicio de la red residencial creado para el dispositivo seleccionado, para eliminarlo en tal caso. A continuación, se diferencia en función de la prioridad seleccionada. Para la prioridad HIGH, el ancho de banda máximo para el dispositivo va a ser igual al contratado, por lo que no es necesario crear ningún servicio adicional, de modo que el ancho de banda disponible se repartirá entre los dispositivos activos de modo equitativo. Sin embargo, para las prioridades MED y LOW, es necesario calcular el máximo ancho de banda que corresponde a cada orden de prioridad, estableciendo este máximo a la mitad del ancho de banda contratado para la prioridad MED y en un tercio para la prioridad LOW. A continuación, se crea el servicio openflow correspondiente. Por último, en todos los casos, se responde afirmativamente al usuario, enviando un mensaje hacia la aplicación web para que se informe por pantalla al usuario de que su petición ha sido concedida.



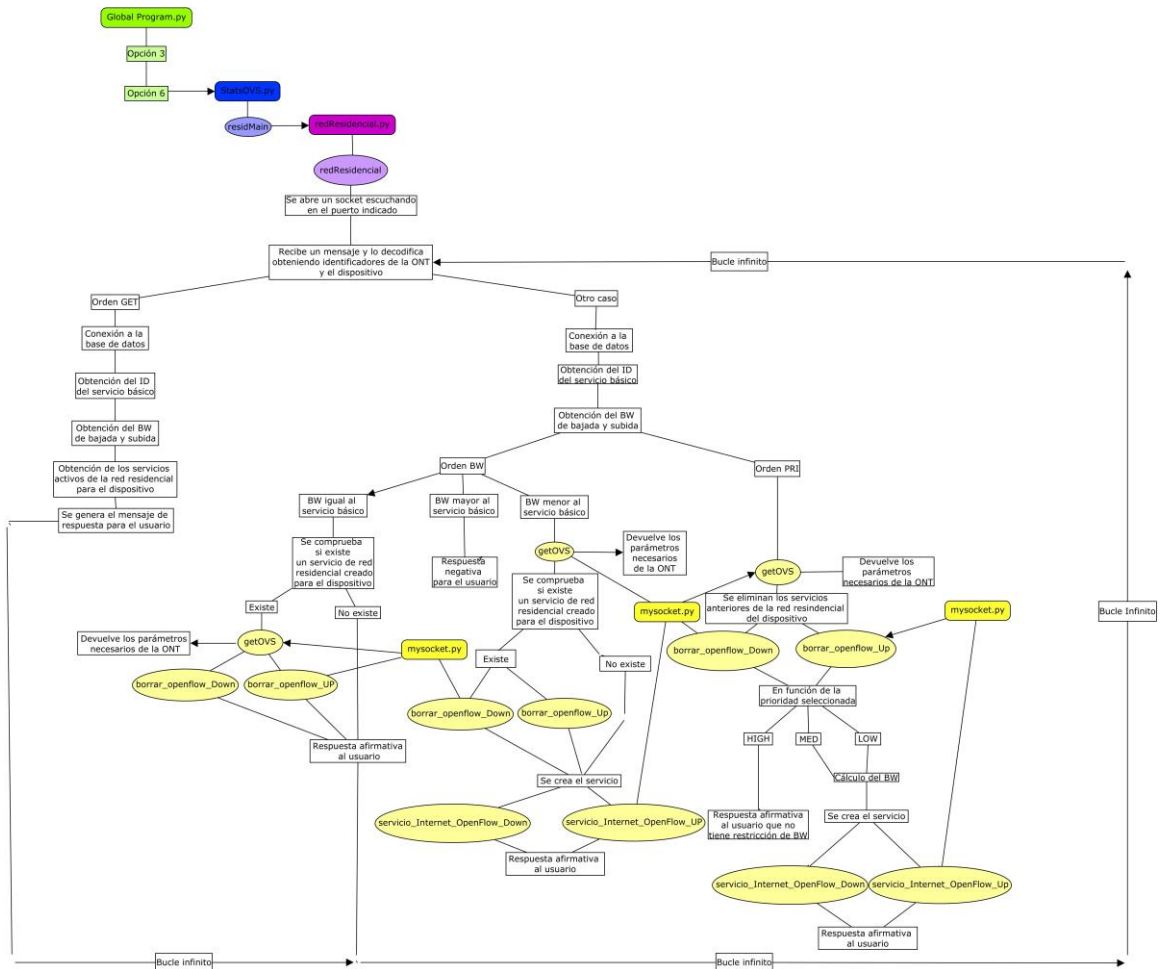


Figura 45: Diagrama de flujo de la aplicación para la gestión de la red residencial

### 4.3 Descripción de las funciones para la automatización de la red SDN-GPON

Después de analizar el funcionamiento de las principales funcionalidades implementadas en el programa global que controla la red SDN-GPON a través de los diagramas de flujo descritos, se ha hecho patente la necesidad de automatizar el código Python, en la medida de lo posible, de modo que aumente su escalabilidad y sea tolerante a cambios de equipos sin necesidad de cambios en el código. Para ello, se han desarrollado algunas funciones que se van a describir en los siguientes puntos.

#### 4.3.1 Automatización del puerto conectado del OLT

Como ya se ha comentado en un apartado previo de esta memoria, existe la posibilidad de usar un puerto PON diferente al puerto PON 0 del OLT, probando en este caso el puerto PON 1, pero pudiendo haber sido cualquier otro de los cuatro puertos

disponibles. Por este motivo, se ha desarrollado una función capaz de detectar el puerto PON del OLT que tenemos conectado para detectar de forma transparente las ONTs asociadas a dichos puertos PON. Telnet, proporciona los comandos necesarios para extraer los identificadores de cada ONT conectada a un puerto PON determinado. Aprovechando estos comandos, mediante un bucle que recorra los diferentes puertos del OLT (0-3), se escribe en un archivo de texto las ONT que tiene conectadas el puerto determinado. Posteriormente, se lee el archivo y se recorre buscando los identificadores de las ONT. En caso de encontrar dichos identificadores, se guarda en una variable el puerto PON para que posteriormente sea devuelto por la función.

Esta función es llamada en aquellos puntos donde necesitamos conectarnos al CLI del OLT, como puedan ser las funciones donde se cargan o se eliminan servicios de una determinada ONT o la función que devuelve una lista con los identificadores de todas las ONTs conectadas al OLT. El código de la función descrita se puede encontrar en el Anexo I.

#### ***4.3.2 Automatización de la obtención de listas con los identificadores de los OVSs, sus interfaces y los puertos que utilizan***

Para poder asociar cada ONT con el identificador del OVS correspondiente se ha desarrollado una función capaz de devolver dos listas, una con los identificadores de los diferentes OVSs de la red y otra con las direcciones MAC de las interfaces de estos OVSs. En primer lugar, a través de una petición HTTP de tipo GET a la url donde se encuentra la interfaz web de nuestro controlador ODL, se extraen los datos de toda la topología y se graban en un archivo JSON. A continuación, se lee dicho archivo y se bucea a través de él hasta encontrar los datos que estamos buscando añadiéndolos en dos listas, una para las direcciones MAC de las interfaces de cada OVS y otra para los identificadores de dichos OVS. Además, aprovechando la lectura del archivo también se extrae una lista con los puertos que utiliza cada OVS, que también es devuelta por la función. Por cada OVS, tenemos tres interfaces, por lo que la lista que contiene las direcciones de las interfaces va a ser tres veces más grande que la lista con los identificadores de los OVS.

Esta función es llamada por la función *getOVS()* que posteriormente devuelve los datos necesarios de la ONT cuando se quiere crear un servicio. En dicha función, tenemos la dirección MAC de la interfaz del OVS que realiza la petición, por lo que mediante un bucle recorreremos la lista buscando la coincidencia y cuando la encontramos, dividimos el índice de la lista entre tres, quedándonos con la parte entera de la división para acudir a la lista con los identificadores de los OVSs. A modo de ejemplo si la dirección MAC es *78:3d:5b:02:e9:cd*, el identificador del OVS que le corresponde es *132204915255757*.

Otras funciones que llaman a esta función son, por un lado, las funciones encargadas de crear los flows por defecto que necesitan los OVS para poder funcionar correctamente (ya que es necesaria la lista con los OVSs) y la función que devuelve la subred a la que pertenece la ONT que está realizando la petición, puesto que necesita la lista con los puertos de cada OVS y la lista con los identificadores de los OVS. El código de la función descrita se puede encontrar en el Anexo I.

### ***4.3.3 Automatización de la asociación de la dirección MAC de la interfaz del OVS con el identificador de la ONT***

Se trata de una función sencilla, que recibe como dato la dirección MAC de una de las interfaces de un OVS y devuelve el identificador que TGMS da a la ONT a la que corresponde dicho OVS. En nuestro programa la dirección de la interfaz que va a recibir la función va a ser siempre la de *br0*, aunque funciona igualmente con cualquiera de las otras dos direcciones de interfaces.

Como ya se ha comentado en otro punto de esta memoria las direcciones MAC de las interfaces de los OVS siguen un patrón y de eso nos vamos a aprovechar en esta función. En primer lugar, la función se queda con los últimos dígitos de la dirección MAC recibida, sustituyendo, además, los “:” de separación entre cada dos dígitos por guiones. Posteriormente, al principio de la dirección que tenemos ahora, se añade “54-4c-52-49-“, común para el identificador de todas las ONTs y es introducido por el fabricante como identificador. De este modo, ya contamos con un identificador con 16 dígitos como los que proporciona TGMS. A continuación, se llama a la función *get\_ID\_ONU()*, que devuelve los identificadores de las ONTs que tenemos conectadas al OLT y buscamos en la lista que devuelve el identificador que corresponde sabiendo que coincidirá en todos

los términos menos el último. Una vez que se encuentra el identificador de la ONT correspondiente se almacena en una variable que posteriormente es devuelta por la función. El código que se acaba de describir, se puede ver en la Figura 46 y en el Anexo I. A modo de ejemplo, si la función recibe como dato la dirección MAC *78:3d:5b:02:e9:cd*, devolverá el identificador *54-4c-52-49-5b-02-e9-ca*.

```
def getOnu(mac_downstream):
    mac_aux=mac_downstream[6:]
    mac_aux=mac_aux.replace(':', '-')
    mac_aux='54-4c-52-49-' + mac_aux

    (MAC)=get_ID_ONU() #Devuelve un vector con las direcciones MAC de las ONTs activas
    #Se recorre el bucle buscando la dirección MAC de la ONT que corresponde con la dirección MAC de la interfaz
    for i in MAC:
        if mac_aux[:len(mac_aux)-1]==i[:len(i)-1]:
            MAC_ONU=i
    return MAC_ONU
```

Figura 46: Código de la función *getONU()*

Ésta, es una función muy específica, que se usa únicamente en la clase *mySocket()*, encargada de la gestión de los servicios temporales.

#### **4.3.4 Automatización en la creación de subredes asociadas a los servidores DHCP de las ONTs**

Es conocido que las ONTs de nivel 2 no tienen capacidad de enrutamiento ni de dar direccionamiento, por lo que a diferencia de las ONTs de nivel 3 no dividen la red en subredes. Por este motivo, se presenta la necesidad de que cada ONT de nivel 2 tenga un servidor DHCP propio que se encargue de proporcionar direccionamiento a los dispositivos que se conecten, generando la subred correspondiente con todos los dispositivos conectados a la red residencial del usuario.

Esto hace que sea necesario que la asociación entre cada ONT y su correspondiente subred se haga de forma automática. Para ello, se ha desarrollado una función que recibe como dato el identificador del OVS de una de las ONTs, como por ejemplo *132204915255757* y devuelve la subred que le corresponde, que para el caso del ejemplo sería *192.168.0.0/26*.

En primer lugar, se abre un socket en la interfaz *br0*, que es la interfaz del ordenador central donde se encuentran los OVSs de las diferentes ONTs conectadas y almacenamos en dos listas los puertos y las direcciones de los hosts de cada OVS. Siguiendo con el ejemplo el puerto sería *54058* y la dirección del host *192.168.0.2*. A

continuación, se hace una llamada a la función *getNodeID()*, descrita anteriormente, de modo que obtenemos las listas que necesitamos con los identificadores de los OVSs y los puertos que utilizan. Mediante un bucle, se busca el identificador del OVS que se ha recibido como dato y se guarda en una variable el puerto que le corresponde a dicho identificador. Utilizando otro bucle se busca dicho puerto en la lista que hemos obtenido del socket y almacenamos en una variable la dirección del host que le corresponde.

A partir de la dirección del host correspondiente es fácil obtener la dirección de la subred correspondiente, ya que como se ha descrito anteriormente en esta memoria para establecer la dirección del host de cada OVS se sigue un patrón, cuestión de la que ahora nos aprovechamos. La dirección de la subred está dos direcciones por debajo de la dirección del host, por lo que hacemos esa resta y añadimos al final “/26”, que va a ser común a todas las subredes con las que vamos a contar. Continuando con el ejemplo, la subred obtenida sería *192.168.0.0/26*.

La función que se acaba de describir es utilizada por la función *getOVS()*, encargada de devolver los datos necesarios cada vez que se quiere crear un servicio y por las funciones encargadas de crear los flows por defecto que necesitan los OVS para poder funcionar correctamente. El código de esta función se puede encontrar en el Anexo I.

#### **4.3.5 Creación de los flows por defecto en el controlador**

Para que los OVSs puedan funcionar correctamente tienen que tener configurados previamente una serie de flows por defecto cuando se inicializa el sistema de control SDN sobre la red GPON. En el OVS del ordenador central, únicamente hace falta un único flow, encargado de pasar a la Tabla 1 en caso de no encontrar nada en la Tabla 0. Mientras que en el caso de cada uno de los OVSs de las ONTs necesitamos configurar dos, uno con la misma función que el que se configura en el OVS del ordenador central, pero con las particularidades propias del correspondiente OVS y otro encargado de evitar que el servidor DHCP instalado en el ordenador central sea quién asigne direccionamiento IP a los equipos conectados más allá del OVS de la ONT. De este modo, será el servidor DHCP instalado en la máquina virtual del OVS quien asigne este direccionamiento a los dispositivos conectados a dicha ONT de nivel 2.

Las funciones encargadas de crear los flows que se han descrito anteriormente, van a ser llamadas al inicio del programa que gestiona la red SDN-GPON, para que, de

esta forma, se eliminan posibles flows indeseados que queden de usos anteriores, evitando de esta forma que se carguen al hacer uso de la red de nuevo. Para ello en primer lugar se llama a la función *getNodeID()*, que nos devuelve una lista con los identificadores de todos los OVSs activos en la red. Haciendo uso de ella, utilizamos un bucle que elimine de la Tabla 0 todos los flows a través de peticiones HTTP de tipo DELETE dejando la Tabla 1, que es donde se encuentra configurado el flow correspondiente al servicio básico.

A continuación, mediante otra petición HTTP, pero en este caso de tipo PUT, configuramos para el OVS del ordenador central el flow encargado de pasar a la Tabla 1, con las especificaciones correspondientes para dicho OVS.

Por último, queda configurar los flows correspondientes a los diferentes OVSs de cada una de las ONTs. Para ello, utilizamos un bucle que recorra la lista que tenemos de todos los OVSs, excluyendo al OVS del ordenador central. El flow encargado de priorizar el servidor DHCP de la ONT sobre el servidor DHCP del ordenador central es igual para todos los OVSs; sin embargo, para el flow encargado de pasar a la Tabla 1 tenemos que distinguir la subred de cada ONT, por lo que necesitamos llamar a la función *getSubred()*, para posteriormente utilizar la petición HTTP de tipo PUT. El código de estas funciones descritas se puede encontrar en el Anexo I.

#### **4.3.6 Funcionamiento de la red residencial utilizando varias ONTs simultáneamente**

Con el objetivo de probar que todas las funcionalidades de automatización desarrolladas funcionan correctamente se ha realizado una prueba empleando dos ONTs diferentes de modo que estén activos simultáneamente los OVSs correspondientes. Y para ello se probarán diferentes funcionalidades de la red residencial en ambas ONTs usando la aplicación web implementada para llevar a cabo la gestión de los dispositivos y servicios de la red residencial. Cabe destacar que estas funcionalidades, correspondientes a la Opción 5 *Residential Network Management*, se han unido en una única opción del programa global de gestión de la red residencial como se puede observar en la Figura 47, pero manteniendo un socket diferente para cada funcionalidad.

```
You have chosen: OpenFlowConfig
1. Attach OpenFlow service to ONT
2. Detach OpenFlow service from ONT
3. Show active OVS
4. Automatic Service Configuration Policies
5. Residential Network Management
6. Go back
What do you want to do?: 
```

Figura 47: Menú Openflow modificado

Los OVSs han sido configurados tal y como se describe en el apartado correspondiente de esta memoria. Para la primera ONT, se va a contratar un servicio temporal por una duración de 10 minutos con 180 Mbps, siendo el servicio básico de 150 Mbps. Para ello, en un ordenador conectado al ordenador donde se encuentra el OVS, que efectúa el papel de cliente, accedemos en un navegador web a la dirección donde se encuentra la aplicación web, en este caso la 192.168.0.2 y contratamos el servicio temporal como se puede ver en la Figura 48.

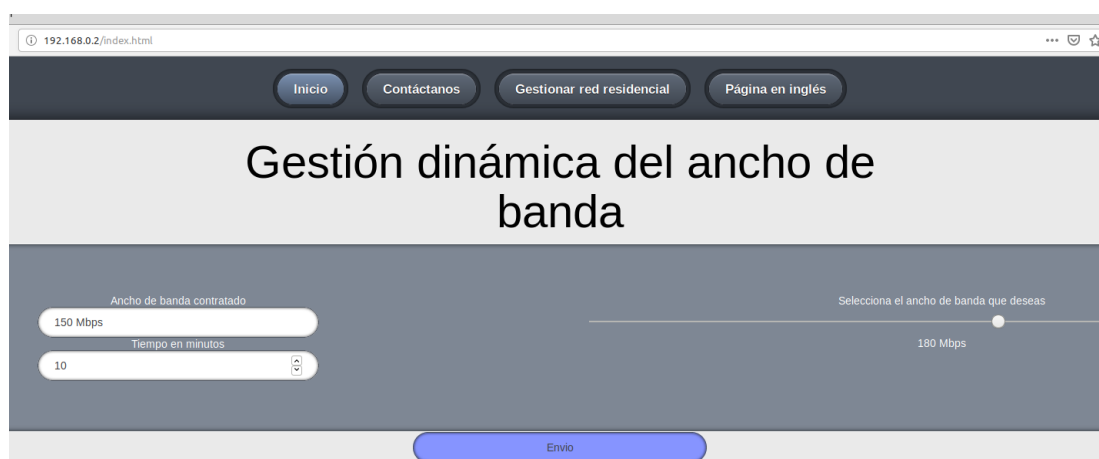


Figura 48: Contratación de un servicio temporal

Una vez contratado dicho servicio temporal y para comprobar si dicho servicio ha sido contratado correctamente, desde nuestro servidor, que es un ordenador conectado al ordenador central y configurado como en otras pruebas que se han explicado anteriormente a lo largo de la memoria, transmitimos a través de *iperf* a una tasa de 200 Mbps obteniendo el resultado que se muestra en la Figura 49. Como se puede observar en la captura de Wireshark recogida en el ordenador que actúa como cliente y situado en la red residencial del usuario (lado de la ONT), el flow está correctamente configurado, ya que al receptor únicamente llegan 180 Mbps.

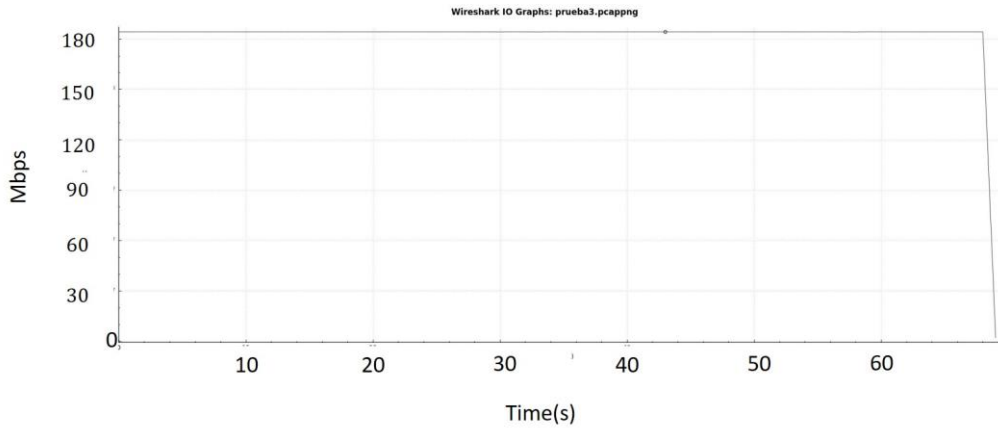


Figura 49: Captura de Wireshark del servicio temporal contratado

Mientras el servicio temporal que hemos contratado se mantiene activo en una ONT concreta, en la otra ONT de nivel 2 que tenemos conectada vamos a establecer una restricción de ancho de banda máximo a uno de los dispositivos registrados en su red residencial, que en este caso se trata de un ordenador conectado al ordenador donde tenemos instalado el OVS y cuyo nombre es *tfglab7-System-Product-Name*, como se puede ver en la Figura 50. Al igual que en la otra ONT, el servicio básico es de 150 Mbps, y se va a establecer una restricción de ancho de banda máximo de 80 Mbps. Para ello, en el ordenador que hemos conectado al ordenador donde se encuentra el OVS, accedemos, en un navegador web, a la dirección donde se encuentra la aplicación web para esta ONT, 192.168.0.66 y establecemos la restricción para el dispositivo como se puede ver en la Figura 51.

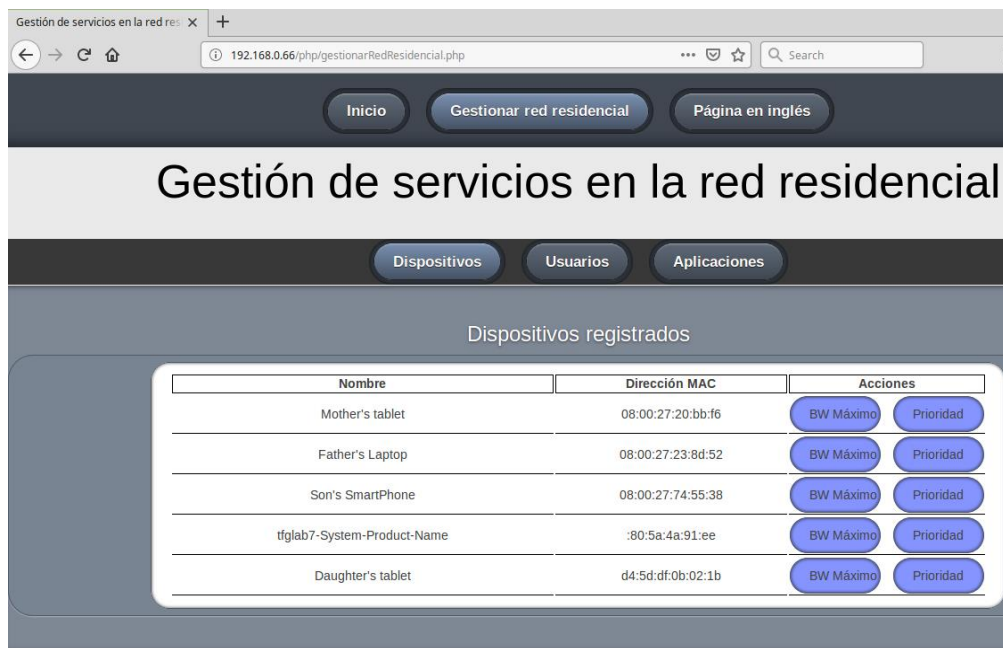


Figura 50: Dispositivos registrados en la aplicación web





Figura 51: Restricción de un ancho de banda para un dispositivo

Como en el caso anterior, desde nuestro servidor, que ya tenemos conectado al ordenador central y configurado de la transmisión anterior, transmitimos mediante *iperf* a una tasa de 200 Mbps. Como se puede ver en la captura de Wireshark recogida en el ordenador que actúa como cliente mostrada en la Figura 52, en esta ocasión el flow también se ha configurado correctamente, ya que el ordenador al que se le ha impuesto la restricción sólo llegan 80 Mbps. Este servicio es simétrico, por lo que, aunque no se muestre, el flow también queda configurado en el sentido de upstream, de modo que no podría transmitir a una tasa superior a 80 Mbps.

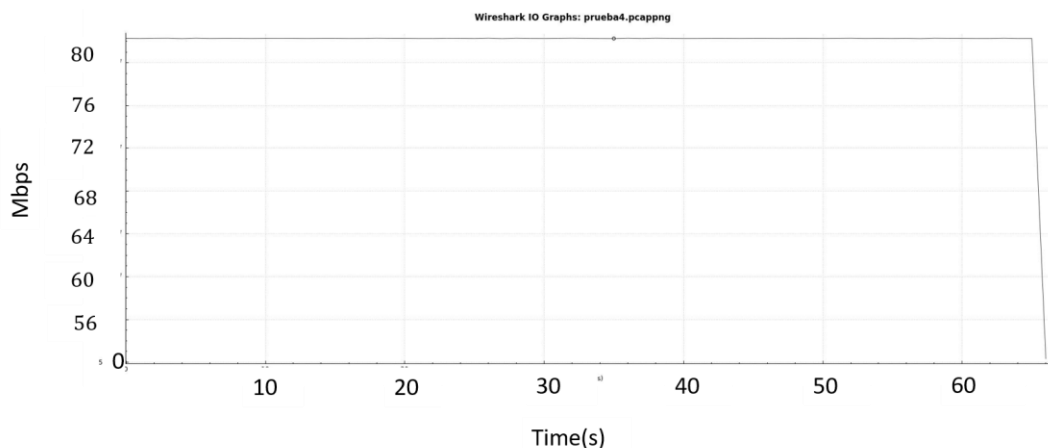


Figura 52: Captura de Wireshark de la restricción de ancho de banda a un dispositivo de la red residencial

Queda demostrado, por lo tanto, que todas las funciones que se han desarrollado con el objetivo de automatizar la red, cumplen perfectamente su función.

## 4.4 Análisis del comportamiento de políticas de configuración de servicios en tiempo real

En un punto anterior de esta memoria, se analizó mediante un diagrama de flujo el comportamiento teórico del algoritmo de configuración dinámica de servicios bajo el caso de uso de tarifas flexibles. Este sistema de tarifas flexibles gestionadas mediante SDN presupone un nuevo modelo de negocio para los operadores, pues aseguran un ancho de banda básico fijo a los usuarios y un ancho de banda extra que solamente darán cuando éstos lo demanden, de modo que mientras no lo demanden los recursos de la red quedarán liberados para poder ser usados de forma transparente entre otros usuarios que tengan mayores necesidades de ancho de banda, de modo que se tiene un mayor control sobre el ancho de banda total disponible para distribuirlo en tiempo real y de forma eficiente entre todos los usuarios aprovechando así al máximo todos los recursos de la red. Además, este tipo de modelos, también resultarán interesantes cuando los proveedores de servicio ofrezcan parte de sus recursos a entidades públicas, como puedan ser ayuntamientos, para, por ejemplo, dar cobertura móvil a los ciudadanos en zonas estratégicas de la ciudad, creando así un modelo de tarificación flexible y dinámica en función de la demanda en tiempo real de los ciudadanos hasta alcanzar un ancho de banda máximo estipulado en función del contrato firmado.

En este punto de la memoria se va a analizar su comportamiento desde un punto de vista más práctico mediante el análisis de las gráficas obtenidas a través de capturas de Wireshark. Para ello, inicialmente vamos a suponer a un usuario que tiene contratada una tarifa flexible con un servicio básico de 150 Mbps simétricos más un 30% de ancho de banda extra, lo que supone que en caso de que el cliente demande un ancho de banda superior al básico, se le concederá, hasta un máximo de 195 Mbps siempre y cuando la red disponga de dichos recursos. En primer lugar, se va a analizar el comportamiento del algoritmo modificando el tamaño de la ventana deslizante que utiliza. En la Figura 53, podemos ver el comportamiento del algoritmo cuando el tamaño de la ventana es de 20 muestras, en la Figura 54 cuando el tamaño de la ventana es igual a 15 muestras y por último en la Figura 55 cuando el tamaño de la ventana es igual a 10 muestras.

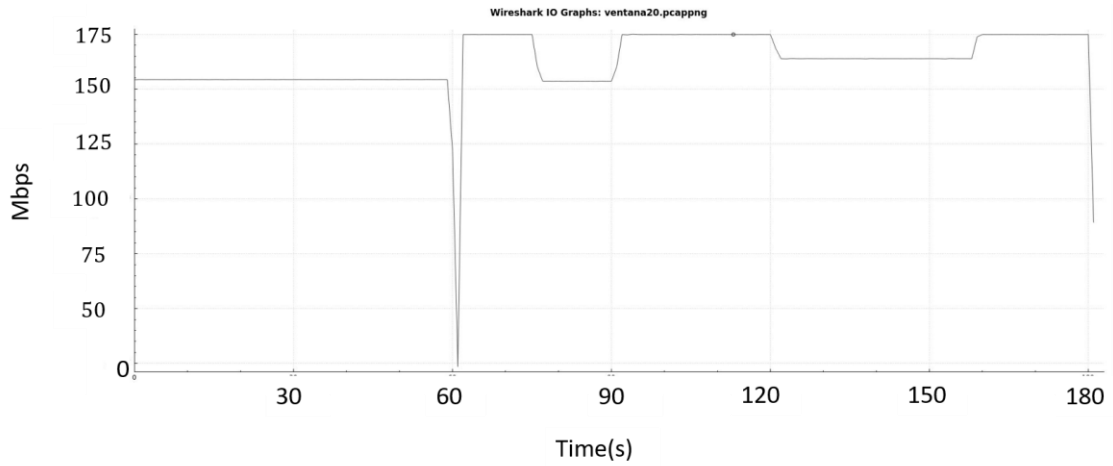


Figura 53: Comportamiento del algoritmo cuando el tamaño de ventana es igual a 20 muestras.

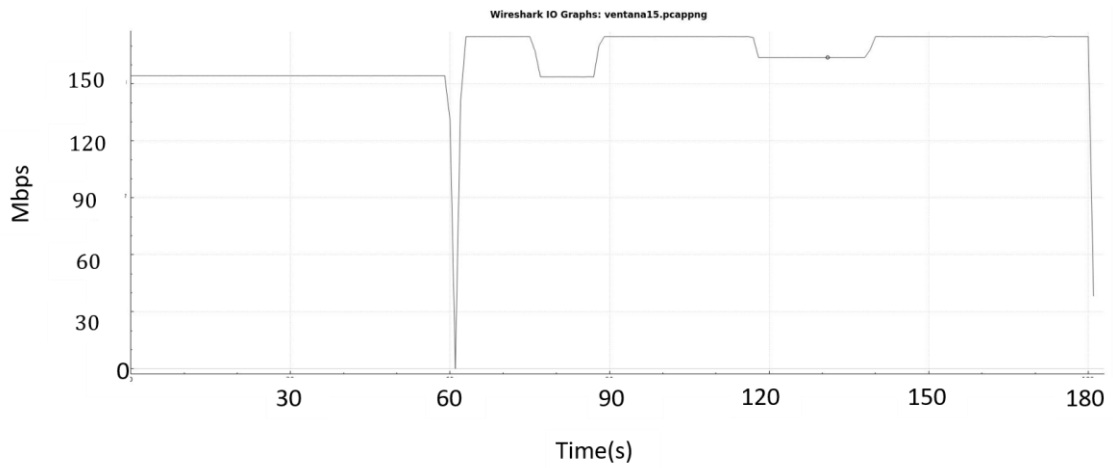


Figura 54: Comportamiento del algoritmo cuando el tamaño de ventana es igual a 15 muestras.

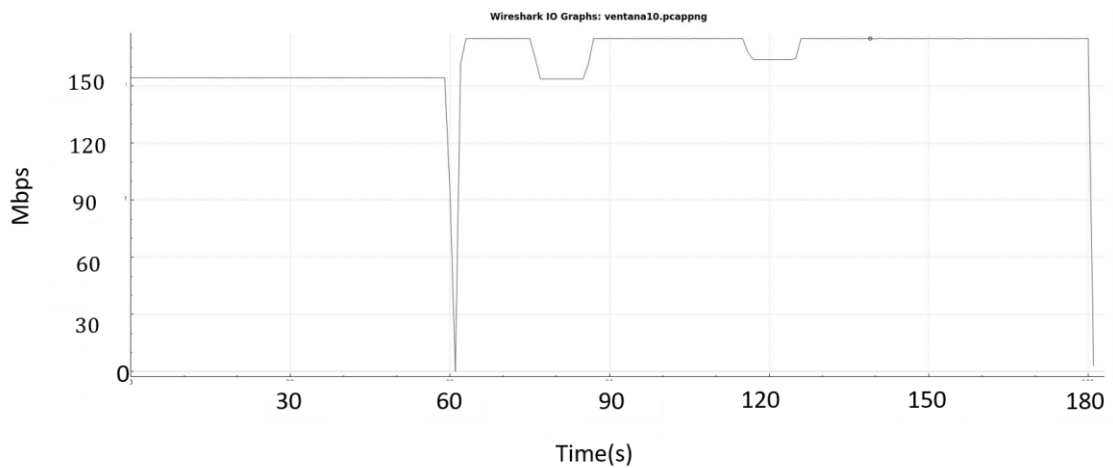


Figura 55: Comportamiento del algoritmo cuando el tamaño de ventana es igual a 10 muestras.

En los tres casos las transmisiones se han llevado a cabo del mismo modo para que a simple vista se puedan observar las diferencias entre los diferentes casos. En primer lugar, se transmite durante un minuto aproximadamente a una tasa de 150 Mbps

que en este caso coincide con el servicio básico y a continuación durante los dos minutos siguientes se transmite a una tasa de 170 Mbps. Como ya es conocido, el algoritmo recoge el valor del ancho de banda demandado en una ventana deslizante, calculando la media de dicha ventana en tiempo real. En el momento en el que detecta que el ancho de banda medio de la ventana es superior al valor actual, espera tres ciclos para comprobar que el ancho de banda medio demandado está por encima de dicho valor, con la finalidad de corroborar que no son fluctuaciones aleatorias en el tráfico demandado. Hecho esto, se procede a modificar el ancho de banda ofrecido en función del servicio contratado redondeando dicho nuevo valor a la decena superior del demandado, por ejemplo, si la media de la ventana se sitúa en 156 Mbps, el algoritmo redondea a 160 Mbps. Sin embargo, antes de crear el flow con el nuevo requisito de ancho de banda, el algoritmo comprueba que existen recursos disponibles en la red (ancho de banda en ambos canales) y que el ancho de banda tras el redondeo no es superior al máximo establecido para la tarifa. En este último caso, el ancho de banda concedido se limitará a este máximo de la tarifa. Por este motivo, en los tres casos de las gráficas anteriores, tenemos un tiempo entre 120-160 segundos para la ventana de 20 muestras, entre 115-140 segundos para la ventana de 15 muestras y entre 112-125 segundos para la ventana de 10 muestras, en el que la tasa de transmisión ofrecida por el algoritmo es de 160 Mbps, ya que la media de la ventana se sitúa durante ese tiempo entre 150 Mbps y 160 Mbps; hasta que finalmente el ancho de banda ofrecido al usuario es de 170 Mbps desde el segundo 160 para la ventana de 20 muestras, desde el segundo 140 para la ventana de 15 muestras y desde el segundo 125 para la ventana de 10 muestras, puesto que la media de la ventana ya estará por encima de 160 Mbps a partir de estos tiempos.

Por otro lado, para los tres tamaños de ventana de las gráficas anteriores, se presentan tres escalones diferentes, con dos zonas de transición entre ellos hasta alcanzar la tasa de transmisión deseada, en este caso de 170 Mbps. El primer escalón y el segundo son exactamente iguales en términos de duración para los tres casos de estudio, pero las dos zonas de transición varían su duración en función del tamaño de la ventana, siendo más pequeña dicha transición a medida que disminuye el tamaño de la ventana, cuestión que es perfectamente lógica dado que al reducir el tamaño de la ventana el tiempo necesario para que ésta se llene se reduce de manera proporcional. En vista de que el tiempo hasta conseguir la tasa de transmisión deseada se reduce a medida que se reduce el tamaño de la ventana sin que ello produzca fluctuaciones en el ancho de banda

ofrecido por el algoritmo, se ha optado por mantener un tamaño de ventana igual a 10 muestras para las pruebas posteriores.

El siguiente paso del análisis va a ser intentar encontrar una explicación para los dos escalones que se pueden ver en la Figura 56, en los que el algoritmo muestra un comportamiento no demasiado óptimo.

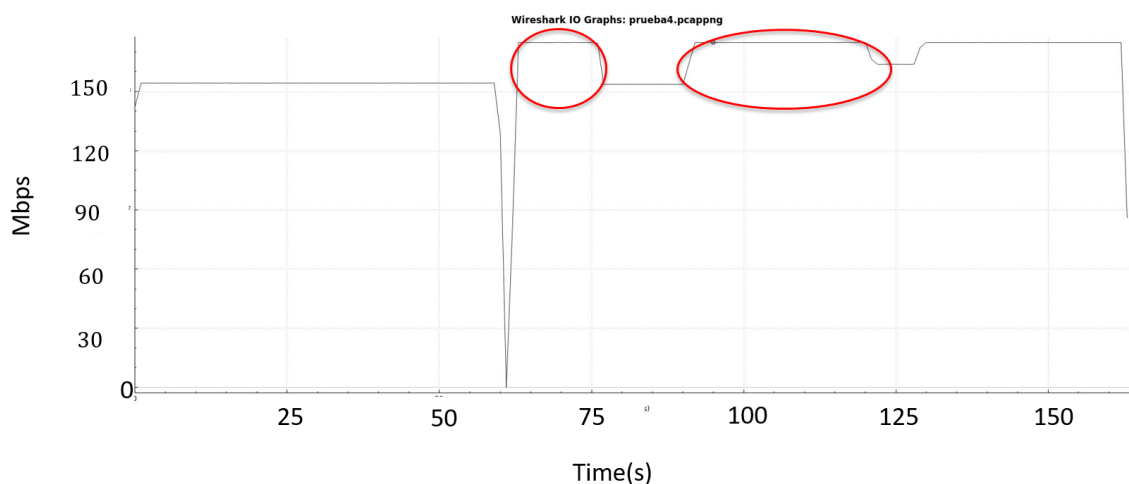


Figura 56: Escalones con un comportamiento inadecuado del algoritmo.

Para el caso del primer escalón, en primer lugar se ha comprobado que el único flow activo durante ese tiempo es el correspondiente al del servicio básico, que en este caso se corresponde con 150 Mbps. Posteriormente, se ha optado por ver el comportamiento del entorno SDN-GPON ante un cambio en la tasa de transmisión enviada por el usuario, transmitiendo durante los primeros 120 segundos a una tasa de 150 Mbps y pasando después a transmitir a 170 Mbps y sin que estén activas ninguna de las funcionalidades de los algoritmos, sino solamente teniendo activo en el OVS un servicio básico, en este caso de 150 Mbps. El resultado respecto al ancho de banda que el OVS concede al usuario con su servicio básico contratado se muestra en la Figura 57. Tal y como se observa, el escalón se mantiene y con una duración similar, por lo que en primer lugar se puede concluir que este comportamiento es ajeno al algoritmo de configuración dinámica de servicios. La causa de este comportamiento reside en el OVS, que es incapaz de detectar instantáneamente cambios en la tasa de transmisión, necesitando un tiempo de reacción de algo más de 10 segundos para detectar un incremento en la tasa de transmisión desde el origen y no dejar pasar más ancho de banda. Esto es, el OVS detecta durante esos 10 segundos que la tasa media sigue siendo

de 150 Mbps, hasta que pasado ese tiempo ya detecta un cambio en la tasa de transmisión a partir de las estadísticas recibidas.

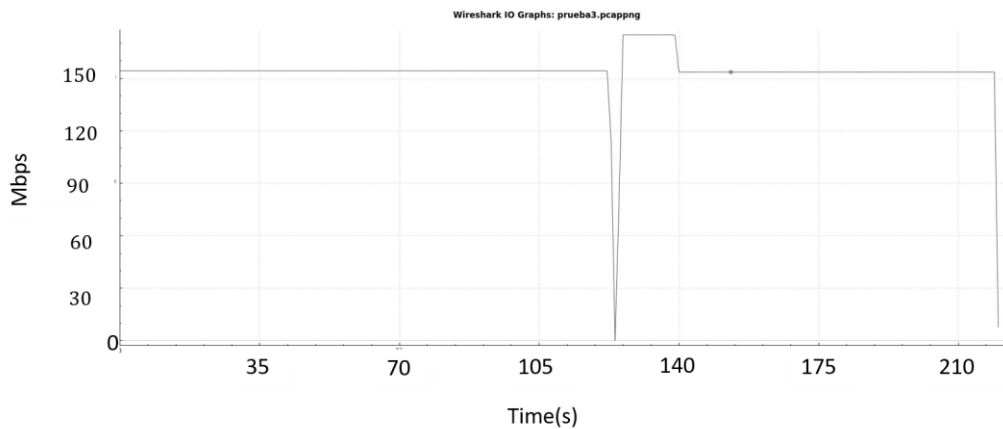


Figura 57: Comportamiento de la red SDN-GPON ante un cambio en la tasa de transmisión.

Para el análisis del segundo escalón, se ha monitorizado al máximo cada paso ejecutado por el algoritmo pudiendo concluir que el inicio del escalón corresponde con el instante en el que se crea un nuevo flow con un nuevo meter asociado a una nueva tasa de transmisión máxima. Para analizar su comportamiento, se van a contratar una serie de servicios temporales a través de la aplicación web del usuario en la red residencial en tiempo real, analizando los resultados obtenidos a través de capturas de Wireshark. La sistemática de las pruebas va a ser la misma en todos los casos, transmitir a una tasa de transmisión de 200 Mbps durante cuatro minutos aproximadamente, que será el tiempo que dure la prueba, partiendo de un servicio básico inicial de 50 Mbps. Tras un minuto de transmisión con dicho servicio básico se procederá a contratar un nuevo servicio temporal durante 2 minutos de 180 Mbps para el caso de la Figura 58, de 150 Mbps para el caso de la Figura 59 y de 100 Mbps para el caso de la Figura 60. Finalmente, durante el último minuto de transmisión, los servicios temporales se eliminan automáticamente, manteniéndose el servicio básico de 50 Mbps. A tenor de los resultados obtenidos, se puede concluir que el OVS necesita un tiempo de reacción tras la creación de un nuevo flow, pero la duración de este tiempo de reacción depende en gran medida de la tasa de transmisión del nuevo flow, viéndose reducido de forma considerable a medida que dichos valores son más bajos.

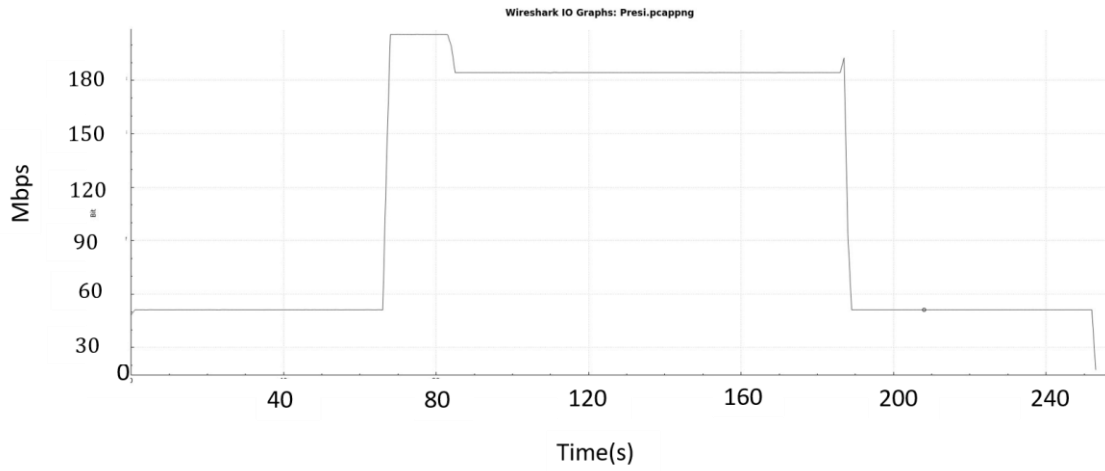


Figura 58: Servicio temporal a una tasa de transmisión de 180 Mbps.

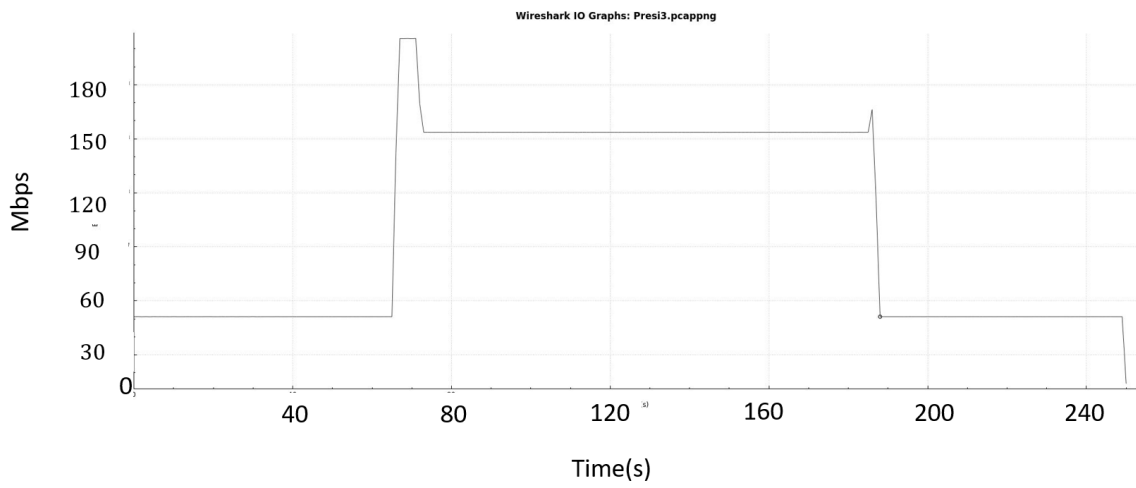


Figura 59: Servicio temporal a una tasa de transmisión de 150 Mbps.

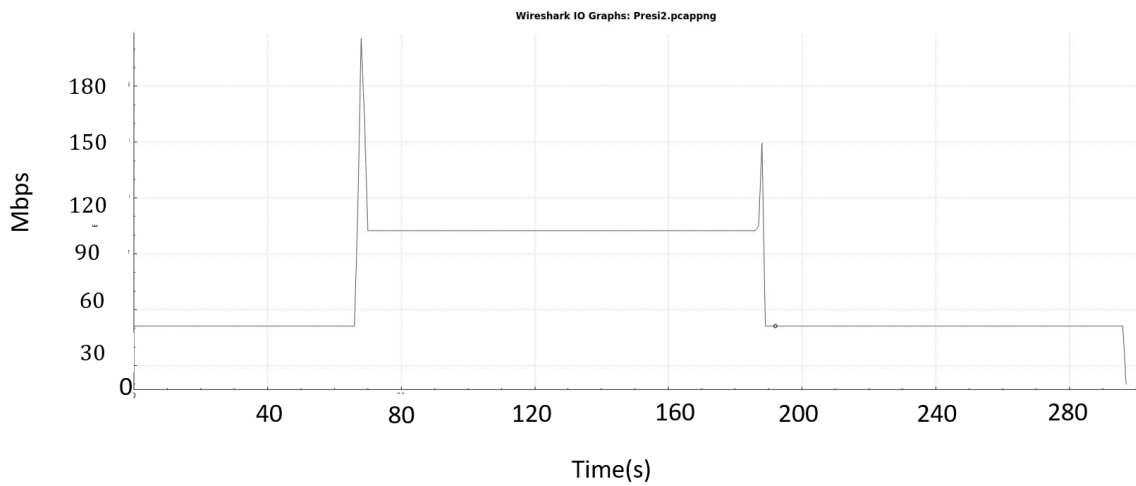


Figura 60: Servicio temporal a una tasa de transmisión de 100 Mbps.

Por último, se va a probar el algoritmo completo, en primer lugar, utilizando las tarifas establecidas anteriormente, para después probar el uso de nuevas tarifas flexibles con diferentes condiciones. Como ya se ha comentado anteriormente el tamaño elegido para la ventana deslizante va a ser de 10 muestras por entender que es la más óptima en tiempo de respuesta para el funcionamiento del algoritmo de configuración dinámica de servicios.

En la Tabla 5, se pueden ver las características de las dos tarifas flexibles que se van a probar en nuestro caso de uso, una tarifa básica con un 20% de extra sobre el ancho de banda básico y una tarifa premium con un 30%. Así, la tarifa básica se aplica cuando el ancho de banda básico contratado es menor a 100 Mbps y la tarifa premium cuando el ancho de banda básico es superior a dicho nivel. Otras tarifas flexibles con diferentes condiciones podrán ser definidas por el operador y/o proveedor de servicios.

Tipo de tarifa flexible	Ancho de banda básico contratado	Ancho de banda en exceso
Tarifa flexible básica	<=100 Mbps	+20% (Mbps)
Tarifa flexible premium	>100 Mbps	+30% (Mbps)

Tabla 5. Tipos de tarifas flexibles definidas para este caso de uso

En la Figura 61 se muestra la prueba realizada para una tarifa premium de 150 Mbps con un ancho de banda extra de 45 Mbps correspondiente al 30%, pudiendo, por lo tanto, alcanzar una tasa de 195 Mbps. Los cambios en la tasa de transmisión demandada por el usuario se producen en saltos de 20 Mbps, manteniendo la misma tasa durante 5 minutos hasta el siguiente cambio, hasta un máximo de 200 Mbps. Por lo tanto, el algoritmo creará nuevos flows en función del ancho de banda medio de la ventana, redondeándolo a la decena superior hasta alcanzar el máximo de 195 Mbps establecido para la tarifa. Durante el primer periodo de cambio en el periodo 60-360 segundos, en el que la tasa de transmisión demandada es de 170 Mbps, el algoritmo presenta tres escalones y dos zonas de transición entre ellos. El primer escalón entre 60-75 segundos, el OVS no es capaz de detectar el cambio en la tasa de transmisión y el valor es de 170 Mbps, seguido por una segunda zona de transición entre 75-90 segundos con valor de 150 Mbps, que es el ancho de banda del flow activo en ese instante y el OVS ya ha reaccionado al cambio en la tasa de transmisión. A continuación, entre 90-120 segundos encontramos el segundo escalón, debido al tiempo de reacción que necesita el OVS ante



la creación de un nuevo flow, que ha sido creado a los 90 segundos, seguido por la segunda zona de transición, entre 120-130 segundos, en los que la tasa de transmisión ofrecida es de 160 Mbps. Por último, a partir de 130 segundos y hasta 360 segundos la tasa de transmisión ofrecida ya alcanza el valor deseado de 170 Mbps. Para el segundo periodo de cambio y el tercero el comportamiento del algoritmo es semejante.

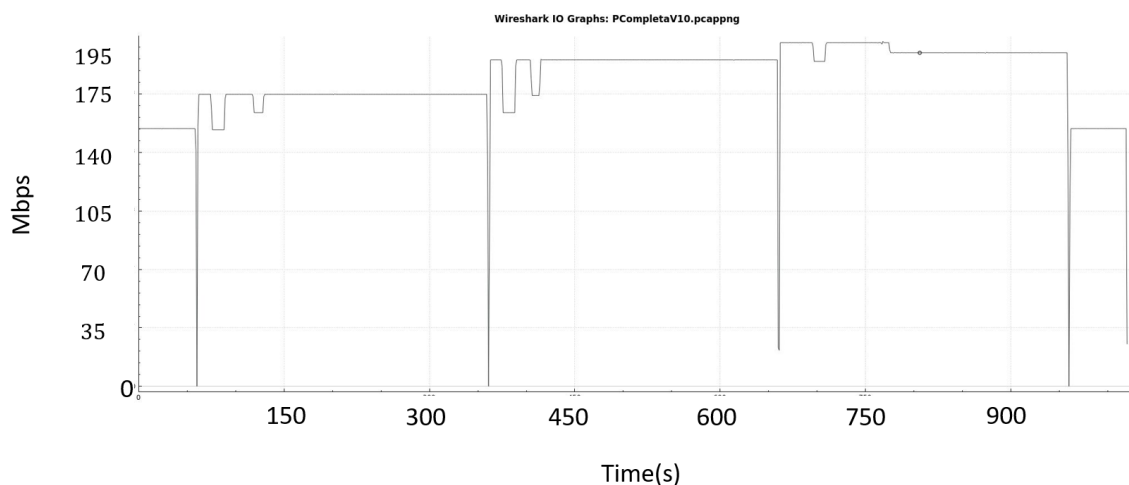


Figura 61: Tarifa básica de 150 Mbps con un ancho de banda flexible asociado a un 30%.

Para el caso de la Figura 62, se utiliza una tarifa flexible de 100 Mbps de ancho de banda básico contratado asociado a un extra de 20 Mbps correspondiente al 20%. Para esta prueba, el incremento de la tasa de transmisión se realizará en saltos de 20 Mbps en periodos de 5 minutos desde los 100 Mbps iniciales hasta un máximo de 140 Mbps (100 Mbps, 120 Mbps, 140 Mbps) y volviendo en el último tramo a transmitir a 100 Mbps. En este caso el algoritmo permitirá al usuario un máximo de 120 Mbps, por lo que la tasa de transmisión ofrecida por el algoritmo será en primer lugar de 110 Mbps, para quedarse estancada en 120 Mbps en el segundo y tercer periodo de la gráfica. De este modo en el segundo periodo de transmisión, a partir del segundo 60, el primer escalón será de valor 120 Mbps entre 60-70 segundos, seguido de un periodo de transición de valor 100 Mbps entre 70-80 segundos. A continuación, un segundo escalón de valor 120 Mbps, entre el segundo 80, instante en el que se crea el flow correspondiente a 110 Mbps, y el 105, seguido por una zona de transición de valor 110 Mbps entre el periodo 105-120 segundos. A partir de ese instante se alcanza la tasa de transmisión de 120 Mbps hasta el segundo 360. En la tercera zona de transmisión, a partir de 360 segundos, solo encontramos un escalón de valor 140 Mbps debido a que el OVS no ha detectado el cambio en la tasa de transmisión. Después, dado que el máximo establecido para la tarifa

es de 120 Mbps, no es necesario crear ningún nuevo flow y la tasa de transmisión ofrecida por el algoritmo se mantiene estable en dicho valor a partir el segundo 375.

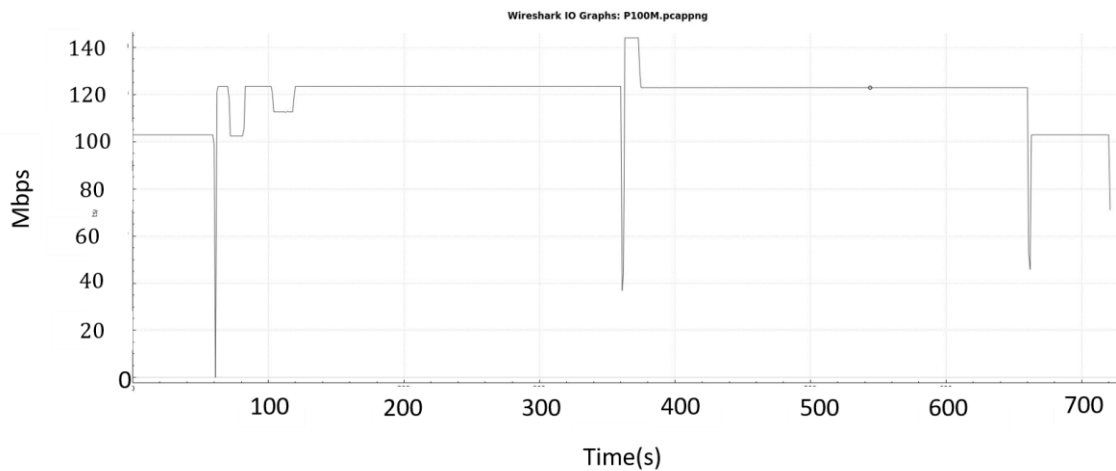


Figura 62: Tarifa básica de 100 Mbps con un ancho de banda flexible asociado del 20%.

Por último, en la Figura 63 se muestra la prueba realizada con la misma tarifa que en el caso anterior, pero en este caso realizando saltos en la transmisión de 10 Mbps cada 5 minutos, esto es, desde 100 Mbps hasta 130 Mbps (100 Mbps, 110 Mbps, 120 Mbps y 130 Mbps) y volviendo finalmente a 100 Mbps. En este caso, dado que el algoritmo redondea a la decena superior del ancho de banda medio demandado en la ventana, solo aparece un único escalón en cada tramo. Así, en el primer tramo que pasa de 100 Mbps a 110 Mbps el algoritmo crea un nuevo flow con 110 Mbps.

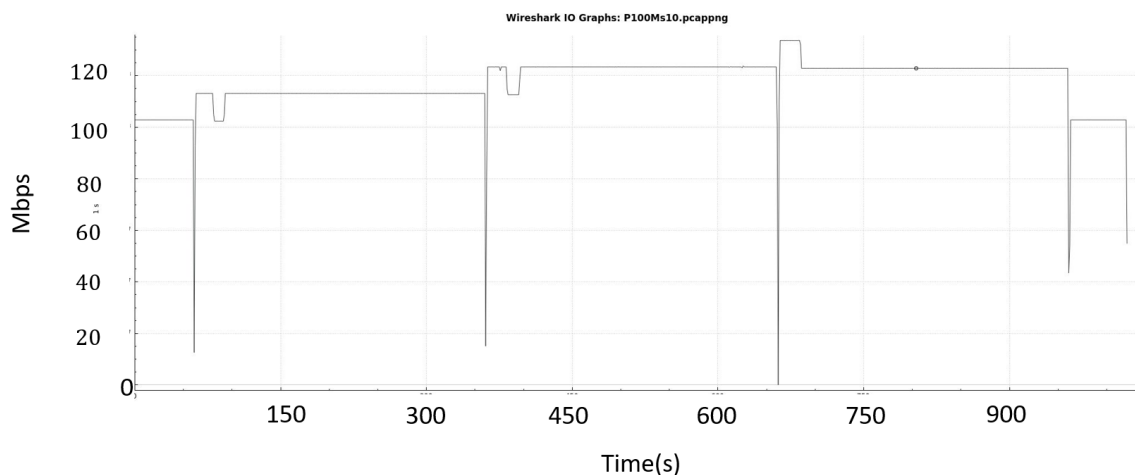


Figura 63: Tarifa de 100 Mbps con una flexibilidad del 20% y saltos de 10 Mbps.

Como ya se ha explicado este escalón que se observa al principio de cada tramo es debido al tiempo de reacción del OVS ante un cambio en la tasa de transmisión, antes de

darse cuenta de este cambio y reaccionar ante el nuevo ancho de banda demandado. Finalmente, también se observa en dicha gráfica que el ancho de banda máximo se sitúa en 120 Mbps que es acorde a la tarifa flexible contratada.

Los resultados obtenidos en todos los casos anteriores concuerdan perfectamente con las conclusiones obtenidas en el análisis de los dos escalones que se ha realizado anteriormente y como podemos ver el tiempo de duración de estos escalones no es significativo ante unos tiempos de transmisión más realistas en un escenario de red y ante un uso real por parte de los usuarios.

A continuación, se van a probar otras nuevas tarifas siguiendo la misma sistemática que se ha seguido para las pruebas anteriores, pero con diferentes características. En la Figura 64 la tarifa flexible se basa en un ancho de banda básico de 100 Mbps con ancho de banda extra del 50%, esto es 150 Mbps en este caso. Para la prueba, los saltos en la tasa de transmisión realizados por el usuario serán de 20 Mbps hasta un máximo de 160 Mbps en periodos de 5 minutos, empezando en 100 Mbps durante aproximadamente el primer minuto y acabando en el último tramo de nuevo en 100 Mbps. En la Figura 64 podemos ver cómo es la evolución del ancho de banda demandado por el usuario en tiempo real durante toda la prueba mediante Wireshark.

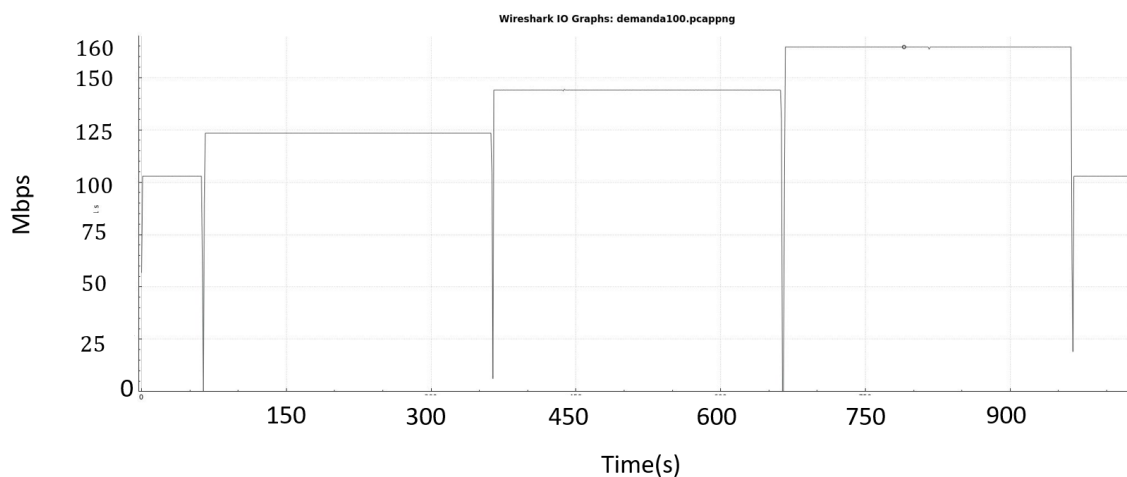


Figura 64: Ancho de banda demandado para la tarifa flexible de 100 Mbps con un ancho de banda extra del 50%.

Tal y como se observa en la Figura 65, a medida que el ancho de banda demandado por el usuario se va incrementando, el algoritmo se va adaptando en tiempo real a esta demanda observándose en el cuarto tramo de transmisión que el ancho de

banda máximo ofrecido al usuario es de 150 Mbps a pesar de que su demanda es de 160 Mbps, ya que el máximo de la tarifa para dicho usuario es este valor.

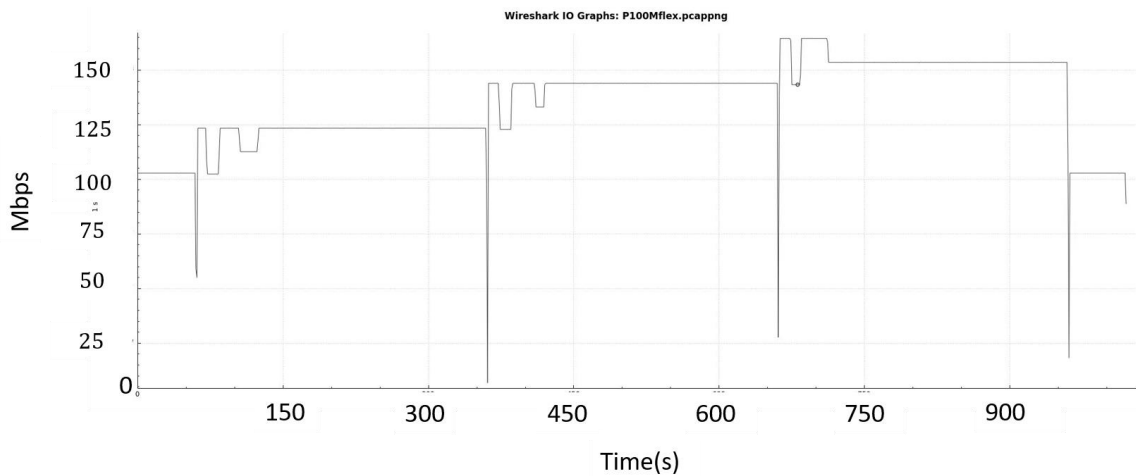


Figura 65: Ancho de banda dado por el algoritmo para una tarifa de 100 Mbps con nivel extra del 50%.

Por último, se va a probar una tarifa flexible de 50 Mbps, pero con un ancho de banda extra del 100%, lo que significa que se puede alcanzar un máximo de 100 Mbps. En la Figura 66, se observa la evolución en tiempo real del ancho de banda demandado realizando saltos en la tasa de transmisión de 20 Mbps en intervalos de 5 minutos desde 50 Mbps hasta un máximo de 110 Mbps, comenzando y terminando transmitiendo a 50 Mbps durante 1 minuto.

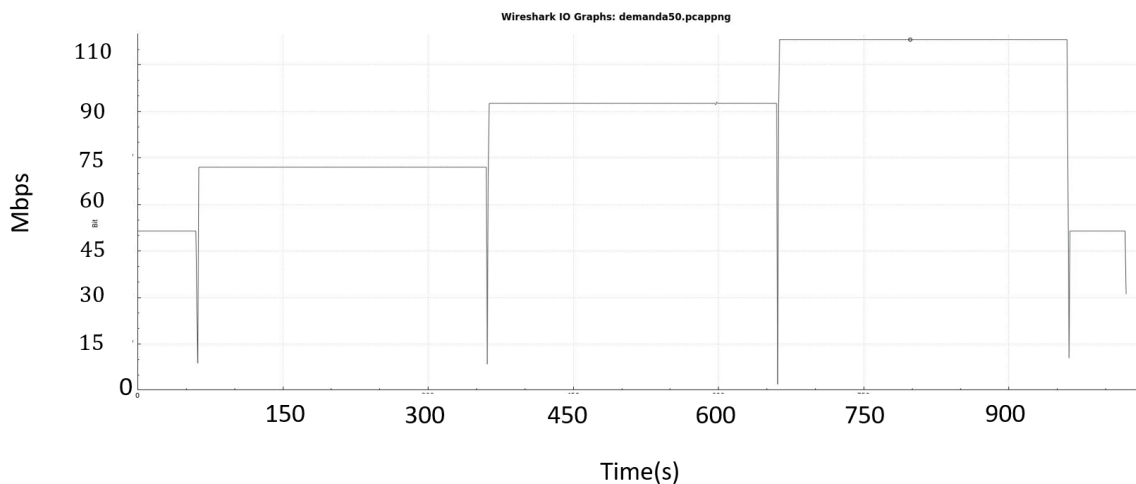


Figura 66: Ancho de banda demandado para la tarifa flexible con un ancho de banda básico de 50 Mbps con un extra del 100%.

El resultado de la evolución en tiempo real del ancho de banda ofrecido por el algoritmo se muestra en la Figura 67. En dicha gráfica se observa que la tasa máxima

ofrecida es de 100 Mbps, alcanzada en el cuarto tramo de transmisión, a pesar de que el máximo ancho de banda demandado es de 110 Mbps en dicho tramo (Figura 66), por lo que el algoritmo se va adaptando en tiempo real al ancho de banda demandado según la tarifa contratada de una forma eficiente y relativamente rápida.

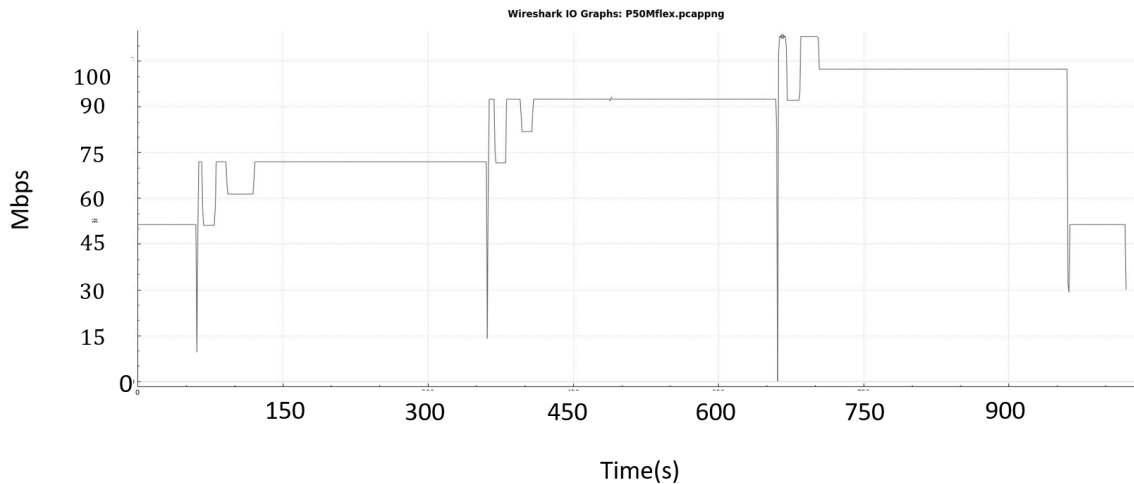


Figura 67: Tarifa flexible con un ancho de banda básico de 50 Mbps con un extra del 100%.

Por último, se han probado dos tarifas en las que el ancho de banda demandado crece muy rápidamente por encima del servicio básico, con el objetivo de emular un escenario con un modelo de negocio diferente, como puede ser una tarifa flexible contratada por un ayuntamiento para dar servicio de internet móvil a una zona determinada de la ciudad, en la que en ciertas franjas horarias el ancho de banda demandado es más elevado por la presencia de más usuarios que en otras franjas en las que la zona presenta una menor demanda. En la Figura 68, se muestra como sería este posible escenario, en el que se muestra una arquitectura combinada de redes Wi-Fi basadas en Radio y Fibra, donde nos encontramos con las ONUs-BS (Optical Network Unit-Base Station) a las que se conectarán los usuarios móviles.

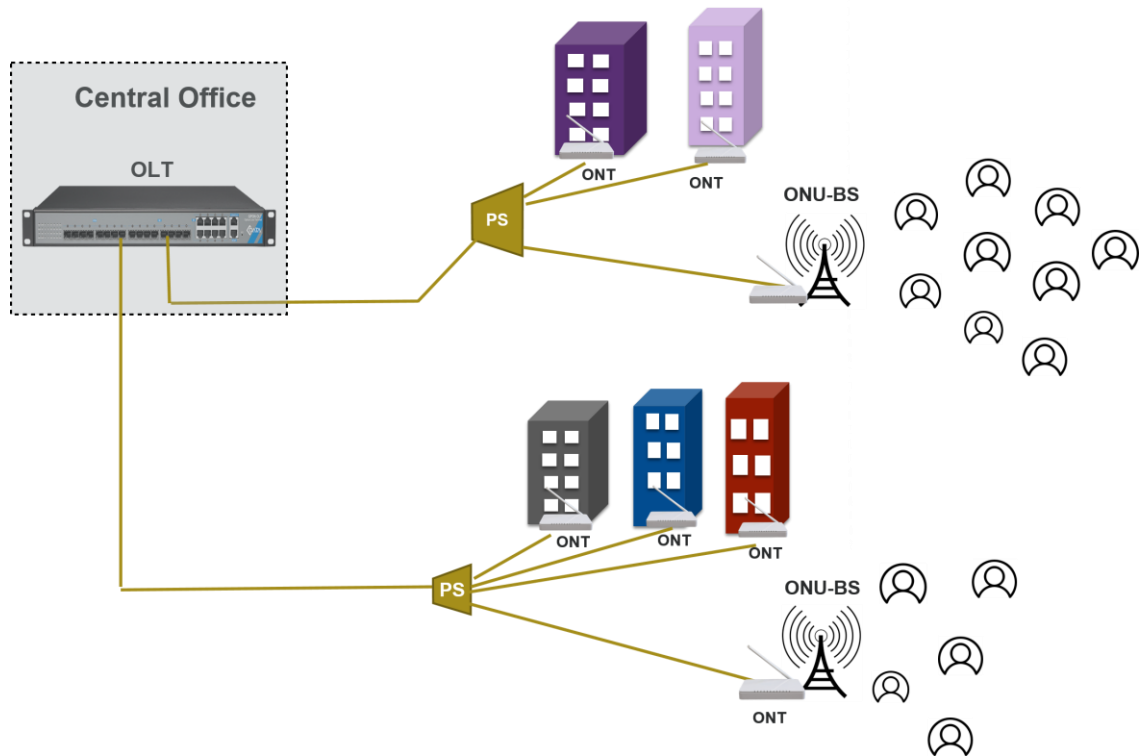


Figura 68: Escenario real para tarifas públicas de carácter flexible

A modo de ejemplo, suponemos un primer escenario (Figura 69) en el que se produce un cambio fuerte en el ancho de banda demandado por la presencia de más usuarios conectados, pasando de una tasa de transmisión demandada de 50 Mbps a 300 Mbps.

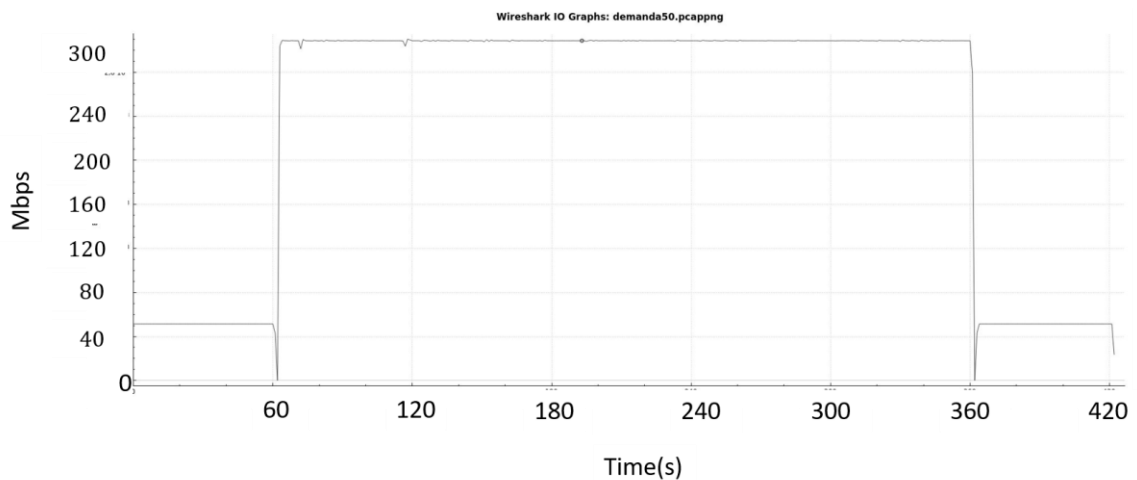


Figura 69: Ancho de banda demandado para la tarifa pública flexible contratada por entidades públicas

Como podemos ver en la Figura 70, que muestra el ancho de banda ofrecido por el algoritmo en tiempo real, el algoritmo reacciona rápidamente ante este gran cambio, ya que la media de la ventana aumenta rápidamente y solo pasa por un escalón intermedio

de 160 Mbps. Esta evolución en tiempo real del valor medio de la ventana se muestra en la Figura 71, donde se observa que en primer lugar permanece estable en torno a 50 Mbps y posteriormente crece de forma más o menos constante hasta alcanzar los 300 Mbps aproximadamente. Así pues, el comportamiento del ancho de banda ofrecido en tiempo real al usuario (Figura 70), es debido a que, como ya se ha comentado anteriormente, el algoritmo concede el ancho de banda redondeando a la decena superior del ancho de banda medio de la ventana, por lo que, si observamos ambas gráficas (Figura 70 y Figura 71), se observa cómo alrededor de los 80 segundos, la media de la ventana tiene un valor aproximado de 155 Mbps y en este momento el algoritmo actualiza el ancho de banda concedido a 160 Mbps.

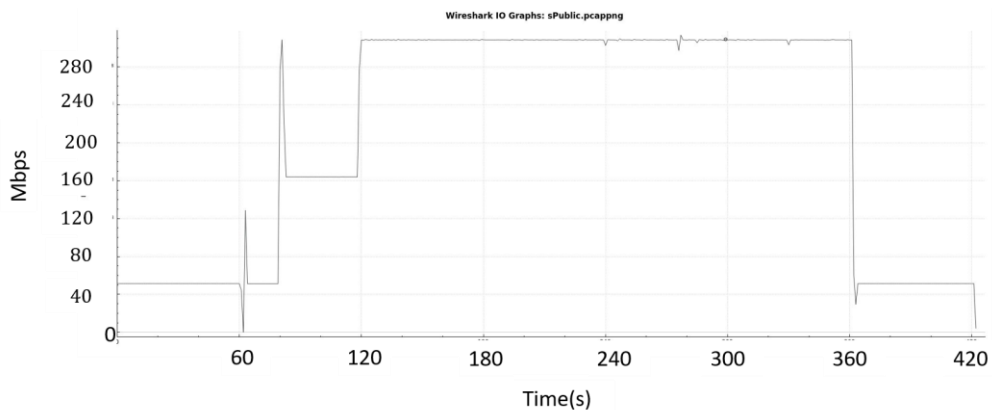


Figura 70: Comportamiento del algoritmo para la tarifa pública flexible contratada por entidades públicas

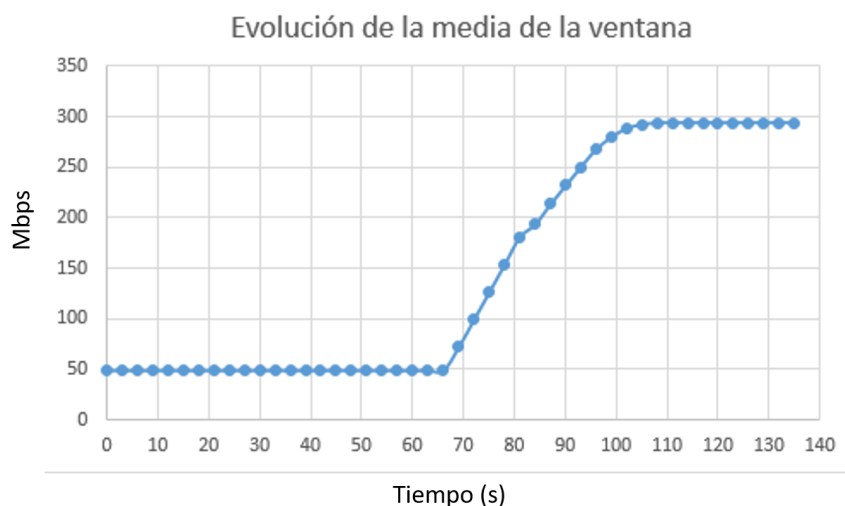


Figura 71: Evolución de la media de la ventana para la tarifa pública flexible

Posteriormente, debido a que el tiempo de las estadísticas OpenFlow que envía el controlador SDN se reinicia al crear un nuevo flow, en este caso el de 160 Mbps, se tiene

que esperar un tiempo a que la ventana se vuelva a llenar y además se completen los tres ciclos de comprobación para crear otro flow acorde al valor medio de la ventana en este momento. En nuestro caso se producirá en torno a los 120 segundos, y el nuevo flow será 300 Mbps que se corresponde con ancho de banda medio de la ventana, tal y como se observa en la Figura 71.

Por último, se va a probar el funcionamiento del algoritmo en un escenario similar con un servicio básico de 100 Mbps, pero pasando de una evolución en tiempo real del ancho de banda demandado desde 100 Mbps hasta 300 Mbps (Figura 72).

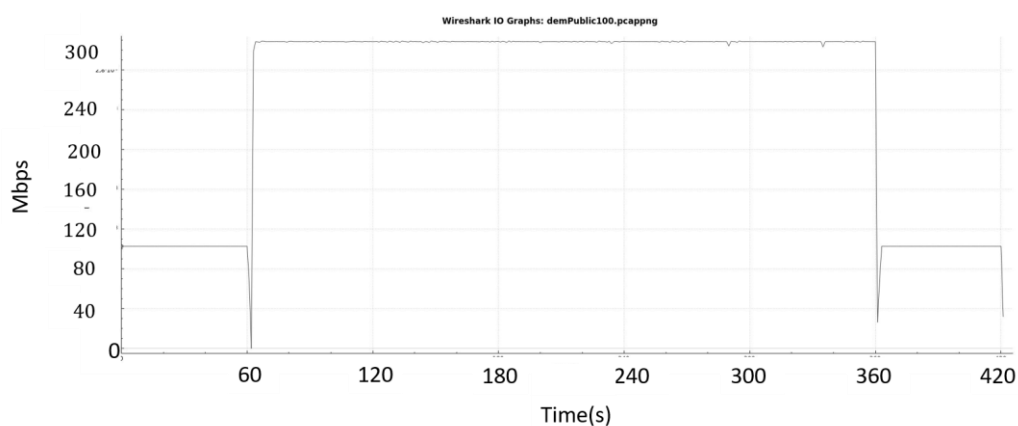


Figura 72: Ancho de banda demandado para la tarifa pública flexible contratada por entidades públicas

El comportamiento del algoritmo, que se muestra en la Figura 73, es similar al del caso anterior, pasando por un único escalón de 190 Mbps en esta ocasión antes de alcanzar la tasa final de 300 Mbps. Además, la evolución de la media de la ventana, que se presenta en la Figura 74 tiene una forma muy similar al caso anterior. La explicación a este comportamiento, la encontramos en las gráficas de la Figura 73 y 74. Alrededor del segundo 80, el ancho de banda concedido por el algoritmo es de 190 Mbps y es acorde con la media de la ventana en ese momento que se sitúa también entorno a ese valor. Posteriormente, como en el caso anterior, hay que esperar a que la ventana se llene de nuevo y pasen los tres ciclos estipulados para que el ancho de banda ofrecido por el algoritmo cambie, que en este caso será en torno a los 120 segundos y crecerá a un nivel de 300 Mbps, ya que el ancho de banda medio de la ventana en ese momento será de dicho valor (Figura 74).



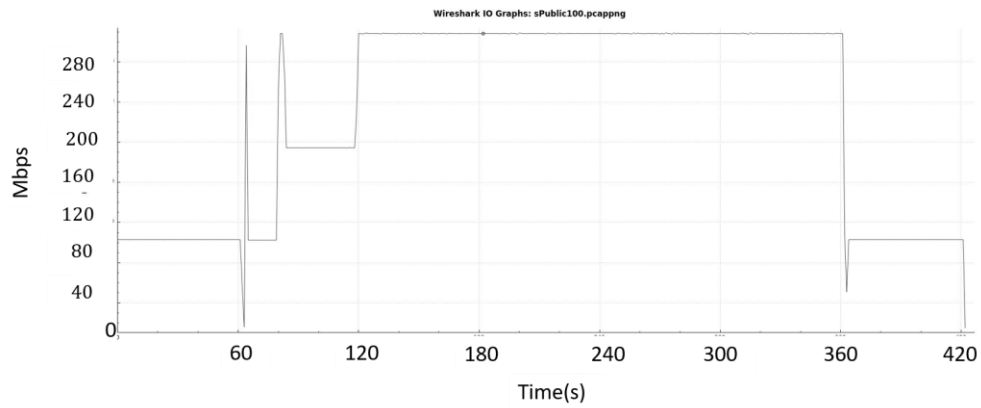


Figura 73: Comportamiento del algoritmo para la tarifa pública flexible contratada por entidades públicas

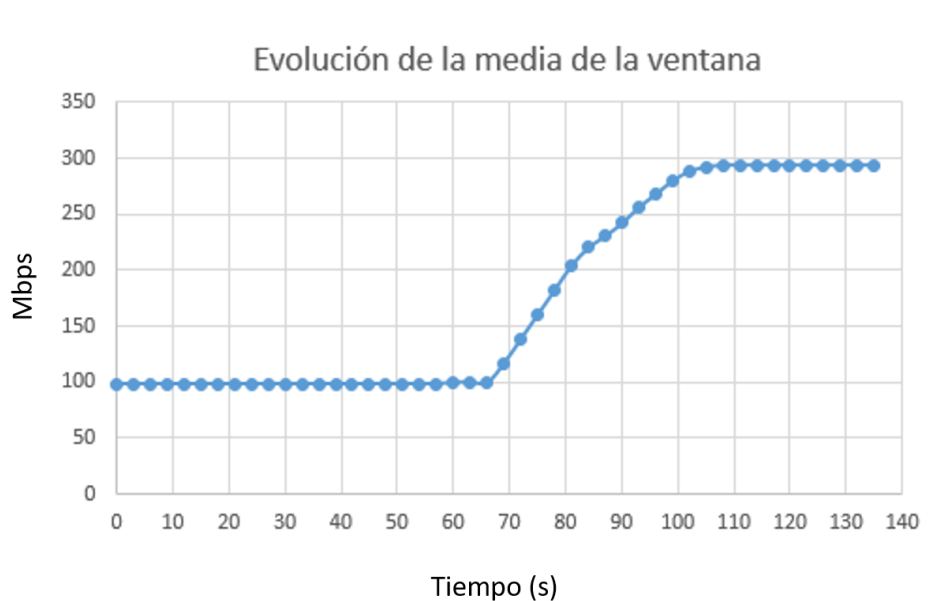


Figura 74: Evolución de la media de la ventana para la tarifa pública flexible

Finalmente, cabe destacar que algunas de estas pruebas del algoritmo de configuración dinámica de servicios también se han llevado a cabo utilizando dos ONTs de nivel 2 conectadas simultáneamente, obteniendo resultados muy similares.

## 4.5 Conclusiones

A lo largo de este capítulo de la memoria se ha llevado a cabo un análisis del funcionamiento de las principales funcionalidades del programa de gestión de la red SDN-GPON mediante la descripción de sus diagramas de flujo para posteriormente desarrollar las funciones necesarias en Python para la automatización de dicho programa. Por último, se ha llevado a cabo una prueba que demuestra el correcto funcionamiento de

dichas funciones. A continuación, se ha hecho un análisis en profundidad del comportamiento del algoritmo de configuración dinámica de servicios dando explicación a aquellos puntos en los que su funcionamiento no es el esperado y realizando un conjunto de pruebas más exhaustivas para su posible aplicación en diferentes casos de uso.

# 5

## Conclusiones y líneas futuras

### 5.1 Conclusiones

En este Trabajo de Fin de Grado, en primer lugar, se han presentado el marco sobre el que después se ha realizado la investigación, describiendo a grandes rasgos las principales características de las redes PON y los fundamentos sobre los que sustenta SDN. A continuación, se han descrito las principales herramientas de software empleadas para el despliegue de SDN sobre la red GPON del laboratorio, tales como son el protocolo OpenFlow, los switches virtuales OVS y el controlador OpenDayLight (ODL). También se ha presentado el lenguaje de programación Python, que es el elegido para desarrollar el programa de gestión global de la red SND-GPON y por último se ha descrito la metodología a seguir para la consecución de los objetivos previamente propuestos.

Por otro lado, se ha descrito el escenario SDN desplegado antes del comienzo de este trabajo, describiendo los principales elementos que lo componen e introduciendo su funcionamiento a grandes rasgos. Posteriormente, se ha hecho un análisis del rendimiento de los dispositivos integrados en la red, encontrando que el throughput entre las tarjetas de red del ordenador central limitaba en gran medida la capacidad de transmisión dentro de nuestro entorno de red desplegado. Por este motivo, se ha realizado los cambios oportunos sobre la topología del ordenador consiguiendo un funcionamiento más óptimo de dicho ordenador y con ello de toda la red SDN-GPON. Una vez hecho esto, se ha podido probar el funcionamiento del puerto PON 1 del OLT y de dos nuevas ONTs de nivel 2, siendo éste muy semejante al comportamiento del puerto PON 0 utilizado hasta el momento y de la ONT de nivel 2 ya desplegada en la maqueta GPON. Dado que las ONTs de nivel 2 necesitan un OVS asociado, ya que no tienen capacidad de enrutamiento, se ha descrito el proceso para la instalación de nuevos OVSs, estableciendo las reglas de configuración oportunas con respecto a las direcciones MAC

de sus interfaces, el servidor DHCP que cada OVS tiene asociado y la aplicación web a la que acceden los usuarios para gestionar y configurar los dispositivos registrados en su red residencial.

En último término, se ha hecho un análisis, a través del diseño y desarrollo de un conjunto de diagramas de flujo de las principales funcionalidades del programa de gestión SDN desarrollado mediante Python, tales como la contratación de servicios temporales por parte del usuario, la gestión de la red residencial de los usuarios que les permite establecer restricciones de ancho de banda sobre dispositivos de su red residencial y la implementación de políticas de configuración dinámica de servicios, que permiten niveles extra de anchos de banda a los usuarios de forma transparente, hasta un máximo establecido en la tarifa contratada, siempre y cuando existan recursos disponibles en la red. A partir de este análisis se han determinado aquellos puntos en los que el código del programa necesitaba ser automatizado para posteriormente desarrollar las funciones necesarias, integrándolas en el código del programa para lograr la automatización deseada. Una vez hecho esto, con el objetivo de que las funciones desarrolladas cumplieran su cometido, se ha probado el funcionamiento de las funcionalidades principales de la red con dos ONTs simultáneamente, viendo que los resultados han sido los esperados. Por último, analizando el comportamiento del algoritmo de configuración dinámica de servicios, se ha podido demostrar que aquellos puntos que no presentan unos resultados iguales a los esperados se deben a que el OVS no es capaz de reaccionar de forma instantánea a cambios en la tasa de transmisión o en la configuración de los flows asociados, de modo que estos comportamientos anómalos se acentúan a medida que se aumenta la tasa de transmisión, siendo sin embargo poco significativos ante escenarios de red más realistas.

## **5.2 Líneas futuras**

Tras la investigación desarrollada en este trabajo y a tenor de los resultados obtenidos, se van a introducir algunas de las vías de estudio que puedan mejorar o agregar funcionalidades a la red SDN-GPON.

Una de las opciones de estudio es la activación, puesta en marcha y prueba de los algoritmos desarrollados utilizando la salida de 10 Gbps que el OLT tiene disponible en lugar de usar las salidas que pertinen tasas de transmisión de 2,5 Gbps para el canal

descendente y 1,25 Gbps para el ascendente. De este modo se podrá analizar el comportamiento de los diferentes dispositivos y de las herramientas software que forman parte de la red SDN-GPON a unas tasas de transmisión mayores.

Por otra parte, en busca de una mayor eficiencia, sería interesante analizar el comportamiento de otros protocolos SDN alternativos a OpenFlow, teniendo como objetivo de nuevo la mejora de las prestaciones de la red SDN-GPON.

Análogamente a la propuesta anterior, después de ver que los OVS instalados presentan un comportamiento cada vez menos eficiente a medida que las tasas de transmisión son cada vez más altas, otro de los puntos de estudio sería analizar otros switches virtuales análogos a OVS, pero que mejoren sus prestaciones en términos de reacción ante cambios a tasas de transmisión altas, ya sean estos cambios en la tasa de transmisión propiamente dicha o en los flows que tienen configurados.

Por último, otra de las posibles vías de estudio, sería el planteamiento y desarrollo de otros modelos de negocio y políticas de asignación dinámica de servicios en la capa SDN, fuera del OLT, del mismo modo que las realizadas en este TFG y los anteriores, con el objetivo de ampliar el abanico de servicios que ya ofrece el escenario de red SDN-GPON desplegado.

# 6

## Bibliografía

- [1] J.C. Ballesta y J. Boltimore «PASSIVE OPTICAL NETWORK (PON): FEATURES AND BENEFITS» [En línea]. Available: <http://www.fundacioniai.org/raccis/v7n2/n13a1.pdf>. [Último acceso: 7 Julio 2020]
  
- [2] A. García Centeno, C. M. Rodríguez Vergel, C. Anías Calderón, F. C. Casmartíño Bondarenko «Controladores SDN, elementos para su selección y evaluación,» [En línea]. Available: <http://revistatelematica.cujae.edu.cu/index.php/tele/article/view/164/153> [Último acceso: 7 Julio 2020]
  
- [3] E. Haleplidis, Ed., K. Pentikousis, Ed., S. Denazis, J. Hadi Salim, D. Meyer, O. Koufopavlou «Software-Defined Networking (SDN): Layers and Architecture Terminology» [En línea]. Available: [https://www.hjp.at/doc/rfc/rfc7426.html#sec\\_3](https://www.hjp.at/doc/rfc/rfc7426.html#sec_3) [Último acceso: 20 Julio 2020]
  
- [4] Open Networking Foundation, «OpenFlow 1.3 Specification,» 25 Junio 2012. [En línea]. Available: <https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf>. [Último acceso: 7 Julio 2020]
  
- [5] OpenDayLight, «OpenDayLight,» [En línea]. Available: <https://www.opendaylight.org/>. [Último acceso: 7 Julio 2020].

- [6] The Linux Foundation, «Open vSwitch» [En línea]. Available: <https://www.openvswitch.org/> [Último acceso: 7 Julio 2020].
- [7] «Página web de Python» [En línea]. Available: <https://www.python.org/> [Último acceso: 7 Julio 2020]
- [8] TELNET Redes Inteligentes, «TELNET-RI,» [En línea]. Available: <http://www.telnet-ri.es>. [Último acceso: 7 Julio 2020].
- [9] «Open Network Operating System (ONOS)» [En línea]. Available: <https://www.opennetworking.org/onos/>. [Último acceso: 14 Julio 2020]
- [10] «iperf - Linux man page,» [En línea]. Available: <https://linux.die.net/man/1/iperf>. [Último acceso: 14 Julio 2020].
- [11] «Adaptador Intel PRO/1000 GT para PC» [En línea]. Available: <https://ark.intel.com/content/www/es/es/ark/products/50480/intel-pro-1000-gt-desktop-adapter.html>. [Último acceso: 14 Julio 2020]
- [12] «Procesador Intel Core i7-9700» [En línea]. Available: <https://www.intel.es/content/www/es/es/products/processors/core/i7-processors/i7-9700.html>. [Último acceso: 14 Julio 2020]
- [13] «Z390 DESIGNARE Placas Base - GIGABYTE» [En línea]. Available: [http://es.gigabyte.com/products/page/mb/z390\\_designare#kf](http://es.gigabyte.com/products/page/mb/z390_designare#kf). [Último acceso: 14 Julio 2020]
- [14] «TG-3468 Adaptador de red Gigabit PCI Express» [En línea]. Available: <https://www.tp-link.com/es/home-networking/adapter/tg-3468/> [Último acceso: 14 Julio 2020]
- [15] «OLT SmartOLT 350» [En línea]. Available: <https://www.telnet-ri.es/olt-gpon-smartolt-350/> [Último acceso: 14 Julio 2020]
- [16] «ont-gpon-carril-din-waveaccess-512» [En línea]. Available: <https://www.telnet-ri.es/ont-gpon-carril-din-waveaccess-512/> [Último acceso:

14 Julio 2020]

- [17] «Oracle VM VirtualBox» [En línea]. Available:  
<https://www.virtualbox.org/> [Último acceso: 14 Julio 2020]



# Anexo I

## Funciones para la automatización del código

### 1. Función para la automatización del puerto del OLT:

```
def get_Puerto():
    # Host y puerto al que se hace la conexión Telnet para acceder al CLI
    host = "172.26.128.38"
    port = "4551"
    # Claves de acceso al CLI
    password1 = "TLNT25"
    password2 = "TLNT145"
    enable = "enable"
    # Acceso al CLI: conexión Telnet al host y puerto indicados anteriormente
    tn = telnetlib.Telnet(host,port,1)
    # Mediante la función write de telnetlib, escritura de los comandos que permiten
    # acceder al menú de privilegios del CLI
    tn.write(password1.encode('ascii') + b"\n")
    time.sleep(0.1)
    tn.write(enable.encode('ascii') + b"\n")
    time.sleep(0.1)
    tn.write(password2.encode('ascii') + b"\n")
    time.sleep(0.1)
    # Bucle que recorre los cuatro puertos del olt para encontrar cual está activo
    channel=0
    puerto = 10
    while channel<=3:
        comandos = "configure \n olt-device 0 \n olt-channel "+str(channel)+"\n show serial-number allocated \n"
        tn.write(comandos.encode('ascii') + b"\n")
        time.sleep(0.1)
        # Lectura de los datos (tanto enviados como recibidos) del CLI y volcado en
        # un fichero de texto para su posterior análisis
        data = tn.read_very_eager().decode()
        # Se abre el fichero con modo de escritura, de esta forma cada vez que cambie
        # el estado de la red el fichero se sobrescribirá con la información nueva
        outfile = open('puertoOLT.txt', 'w')
        # Se escriben los datos procedentes de la escritura de los comandos de arriba
        outfile.write(data)
        # Cierre del fichero
        outfile.close()
        # Se abre el archivo anterior en modo lectura
        outfile = open('puertoOLT.txt', 'r')
        # Se almacenan todas las líneas del fichero con la función readlines()
        lines = outfile.readlines()
        # Bucle que recorre cada línea del fichero
        for line in lines:
            # Se almacenan todas las palabras de cada línea
            palabras = line.split()
            # Bucle que recorre cada palabra de la línea
            for p in palabras:
                # Si los 18 primeros caracteres de una palabra coinciden con los indicados,
                # se trata de una dirección MAC -> Se ha detectado una ONU
                if p[:16]=='54-4c-52-49-5b-0':
                    # Se guarda el puerto.
                    puerto=channel
        # Cierre del archivo
        outfile.close()
        channel+=1
    return puerto
```

## 2. Función para la obtención de listas con los identificadores de los OVSs, sus interfaces y los puertos que utilizan:

```
def getNodeID():
    head = {'Content-type':'application/yang.data+json','Accept':'application/json, text/plain'}
    urlNodes="http://10.0.103.45:8181/restconf/operational/opendaylight-inventory:nodes"
    OVSlist=[]
    MACOVSlst=[]
    Portlist=[]

    r=requests.get(urlNodes,auth=HTTPBasicAuth('admin', 'admin'),headers=head)
    v = r.text
    v=v[9:len(v)-1] #Los datos vienen con un título que es necesario eliminar para poder después buscar en el json

    f = json.loads(v)
    with open('topology.json', 'w') as file:
        json.dump(f, file, indent=4)
    r.raise_for_status()

    x = open("topology.json", "r")
    content = x.read()
    jsondecoded = json.loads(content)
    #Busca en el json los datos que queremos
    for entity in jsondecoded["node"]:
        for interfaces in entity["node-connector"]:
            MACOVSlst.append(interfaces["flow-node-inventory:hardware-address"])

    for entity in jsondecoded["node"]:
        y=entity["id"]
        y=y[9:len(v)-1]
        OVSlist.append(y)

    for entity in jsondecoded["node"]:
        y=entity["flow-node-inventory:port-number"]
        Portlist.append(y)

    #Hay tres direcciones MAC por cada OVS, una por cada interfaz
    return OVSlist, MACOVSlst, Portlist
```

### 3. Función para la asociación de la dirección MAC de la interfaz del OVS con el identificador de la ONT:

```
def getOnu(mac_downstream):

    mac_aux=mac_downstream[6:]
    mac_aux=mac_aux.replace(':', '-')
    mac_aux='54-4c-52-49-'+mac_aux

    (MAC)=get_ID_ONU() #Devuelve un vector con las direcciones MAC de las ONTs activas
    #Se recorre el bucle buscando la dirección MAC de la ONT que corresponde con la dirección MAC de la interfaz
    for i in MAC:
        if mac_aux[:len(mac_aux)-1]==i[:len(i)-1]:
            MAC_ONU=i

    return MAC_ONU
```

### 4. Función para la obtención de las subredes asociadas a los servidores DHCP de las ONTs:

```
def getSubred(ovs_upstream):

    # Se abre un socket escuchando el tráfico TCP en la interfaz br0
    s=socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.getprotobyname('tcp'))
    s.bind(('br0',3))

    port_arr=[]
    host_arr=[]
    cont=0
    i=1
    # Mediante un bucle suficientemente grande buscamos los paquetes que se dirigen hacia los OVS de upstream
    while cont<20:
        pack=s.recv(1024)
        if (str(pack[26])+'.'+str(pack[27])+'.'+str(pack[28])+'.'+str(pack[29]))=='10.0.103.45' and len(pack)>66 and pack[66]==4 and int.from_bytes(pack[34:36],byteorder='big')==6633:
            h=str(pack[30])+'.'+str(pack[31])+'.'+str(pack[32])+'.'+str(pack[33])
            p=int.from_bytes(pack[36:38],byteorder='big')
            if not p in port_arr:
                port_arr.append(p)
                host_arr.append(h)
                i+=1
            cont+=1
    s.close()

    (OVSlst, MACOVSlst, Portlst)=getNodeID()
    i=0
    while i<len(OVSlst):
        if OVSlst[i]==ovs_upstream:
            port=Portlst[i]
            i +=1

    j=0
```

```

while j<len(port_arr):
    if port_arr[j]==port:
        host=host_arr[j]
        j +=1

aux=len(host)-1
v=int(host[aux])
v=v-2

if len(host)>12:
    y=int(host[aux-2])
    x=int(host[aux-1])
    if v<0:
        x=x-1
        v=v+10
        ip_subn="192.168.0."+str(y)+str(x)+str(v)+"/26"
    else:
        ip_subn="192.168.0."+str(y)+str(x)+str(v)+"/26"

elif len(host)==12:
    x=int(host[aux-1])
    if v<0:
        x=x-1
        v=v+10
        ip_subn="192.168.0."+str(x)+str(v)+"/26"
    else:
        ip_subn="192.168.0."+str(x)+str(v)+"/26"

else:
    ip_subn="192.168.0."+str(v)+"/26"

if "192.168.0." not in host:
    ip_subn=None

return ip_subn

```

## 5. Funciones para la creación de los flows por defecto en el controlador:

```

def delete_flows():
    (OVSlist, MACOVslist, Portlist)=getNodeID()
    for OVS in OVSlist:
        head = {'Content-type':'application/yang.data+json','Accept':'application/json, text/plain'}
        urlTable="http://10.0.103.45:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:"+OVS+"/flow-node-inventory:table/0"
        requests.delete(urlTable,auth=HTTPBasicAuth('admin', 'admin'))

```

```

def flow_down():
    head = {'Content-type': 'application/yang.data+json', 'Accept': 'application/json, text/plain'}
    urlTable = "http://10.0.103.45:8181/restconf/config/opensdlight-inventory:nodes/node/openflow:159303472399026/flow-node-inventory:table/0"
    urlDownstream = "http://10.0.103.45:8181/restconf/config/opensdlight-inventory:nodes/node/openflow:159303472399026/flow-node-inventory:table/0/flow/toTable1"

    table = '''
    {
      "table": [
        {
          "id": "0"
        }
      ]
    }
    '''

    r = requests.put(urlTable, auth=HTTPBasicAuth('admin', 'admin'), headers=head, data=table)
    print(r.text)
    r.raise_for_status()

```

```

flowDown = '''
{
  "flow": [
    {
      "id": "toTable1",
      "match": {
        "in-port": "LOCAL",
        "ethernet-match": {
          "ethernet-type": {
            "type": "0x0800"
          }
        },
        "ipv4-destination": "192.168.0.0/24",
        "vlan-match": {
          "vlan-id": {
            "vlan-id-present": "false"
          }
        }
      },
      "instructions": [
        "instruction": {
          "order": "1",
          "go-to-table": {
            "table_id": "1"
          }
        }
      ],
      "flow-name": "toTable1",
      "priority": "7000",
      "idle-timeout": "0",
      "cookie": "478478457845784600",
      "table_id": "0"
    }
  ]
}
'''

r = requests.put(urlDownstream, auth=HTTPBasicAuth('admin', 'admin'), headers=head, data=flowDown)
print(r.text)

```

```

def flow_upl():
    (OVSlst, MACOVSlst, Portlst)=getNodeID()
    for OVS in OVSlst:
        if OVS != "159303472399026":
            (ip_subn)=getSubred(OVS)

            head = {'Content-type':'application/yang.data+json','Accept':'application/json, text/plain'}
            urlTable="http://10.0.103.45:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:"+OVS+"/flow-node-inventory:table/0"
            urlUpstreaml="http://10.0.103.45:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:"+OVS+"/flow-node-inventory:table/0/flow/toTable1"

            table="""
            {
                "table": [
                    {
                        "id": "0"
                    }
                ]
            }
            """

            r=requests.put(urlTable,auth=HTTPBasicAuth('admin', 'admin'),headers=head,data=table)
            print(r.text)
            r.raise_for_status()

```

```

flowUp="""{
    "flow": [
        {
            "id": "toTable1",
            "match": {
                "in-port": "2",
                "ethernet-match": {
                    "ethernet-destination": {
                        "address": "90:e2:ba:e2:42:b2"
                    },
                    "ethernet-type": {
                        "type": "0x0800"
                    }
                },
                "ipv4-source": ""+str(ip_subn)+"",
                "vlan-match": {
                    "vlan-id": {
                        "vlan-id-present": "false"
                    }
                }
            },
            "instructions": {
                "instruction": {
                    "order": "1",
                    "go-to-table": {
                        "table_id": "1"
                    }
                }
            },
            "flow-name": "toTable1",
            "priority": "7000",
            "idle-timeout": "0",
            "cookie": "478478457845784600",
            "table_id": "0"
        }
    ]
}
"""

r=requests.put(urlUpstreaml,auth=HTTPBasicAuth('admin', 'admin'),headers=head,data=flowUp)

```

```

def flow_up2():
    (OVList, MACOVList, PortList)=getNodeID()
    for OVS in OVList:
        if OVS != "159303472399026":
            head = {'Content-type': 'application/yang.data+json', 'Accept': 'application/json, text/plain'}
            urlUpstream2="http://10.0.103.45:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:"+OVS+"/flow-node-inventory:table/0/flow/dropDHCP"

```

```

flowDHCP='''{
    "flow": [
        {
            "id": "dropDHCP",
            "match": {
                "in-port": "2",
                "ethernet-match": {
                    "ethernet-type": {
                        "type": "0x0800"
                    }
                },
                "vlan-match": {
                    "vlan-id": {
                        "vlan-id-present": "false"
                    }
                },
                "ip-match": {
                    "ip-protocol": "17"
                },
                "udp-source-port": "68",
                "udp-destination-port": "67"
            },
            "instructions": {
                "instruction": [
                    {
                        "order": "1",
                        "apply-actions": {
                            "action": [
                                {
                                    "order": "1",
                                    "output-action": {
                                        "output-node-connector": "LOCAL"
                                    }
                                }
                            ]
                        }
                    }
                ]
            }
        }
    ],
}'''

```

```

        "flow-name": "dropDHCP",
        "priority": "50000",
        "idle-timeout": "0",
        "cookie": "478478457845784600",
        "table_id": "0"
    }
}'''

r=requests.put(urlUpstream2, auth=HTTPBasicAuth('admin', 'admin'), headers=head, data=flowDHCP)
print(r.text)
r.raise_for_status()

```